aws

用户指南

Amazon EKS

Amazon EKS: 用户指南

Table of Contents

什么是 Amazon EKS？

Amazon Elastic Kubernetes Service（Amazon EKS）是一项托管服务，无需在 Amazon Web Services (AWS) 上安装、操作和维护自己的 Kubernetes 控制面板。Kubernetes 是一个开源系统，用于自动管理、扩展和部署容器化应用程序。

Amazon EKS 的功能

以下是 Amazon EKS 的主要功能：

安全联网和身份验证

Amazon EKS 将您的 Kubernetes 工作负载与 AWS 联网和安全服务相集成。它还与 AWS Identity and Access Management（IAM）集成，为您的 Kubernetes 集群提供身份验证。

轻松扩展集群

借助 Amazon EKS，您可以根据工作负载的需求轻松扩缩 Kubernetes 集群。Amazon EKS 支持基于 CPU 或自定义指标的水平 Pod 自动扩展，以及基于整个工作负载需求的集群自动扩展。

托管 Kubernetes 体验

您可以使用 `eksctl`、AWS Management Console、AWS Command Line Interface (AWS CLI)、API、`kubectl` 和 Terraform 对 Kubernetes 集群进行更改。

高可用性

Amazon EKS 可为您的控制面板提供跨多个可用区的高可用性。

与 AWS 服务的集成

Amazon EKS 与其他 AWS 服务集成，为部署和管理容器化应用程序提供了一个全面的平台。您还可以使用各种可观测性工具轻松地对 Kubernetes 工作负载进行问题排查。

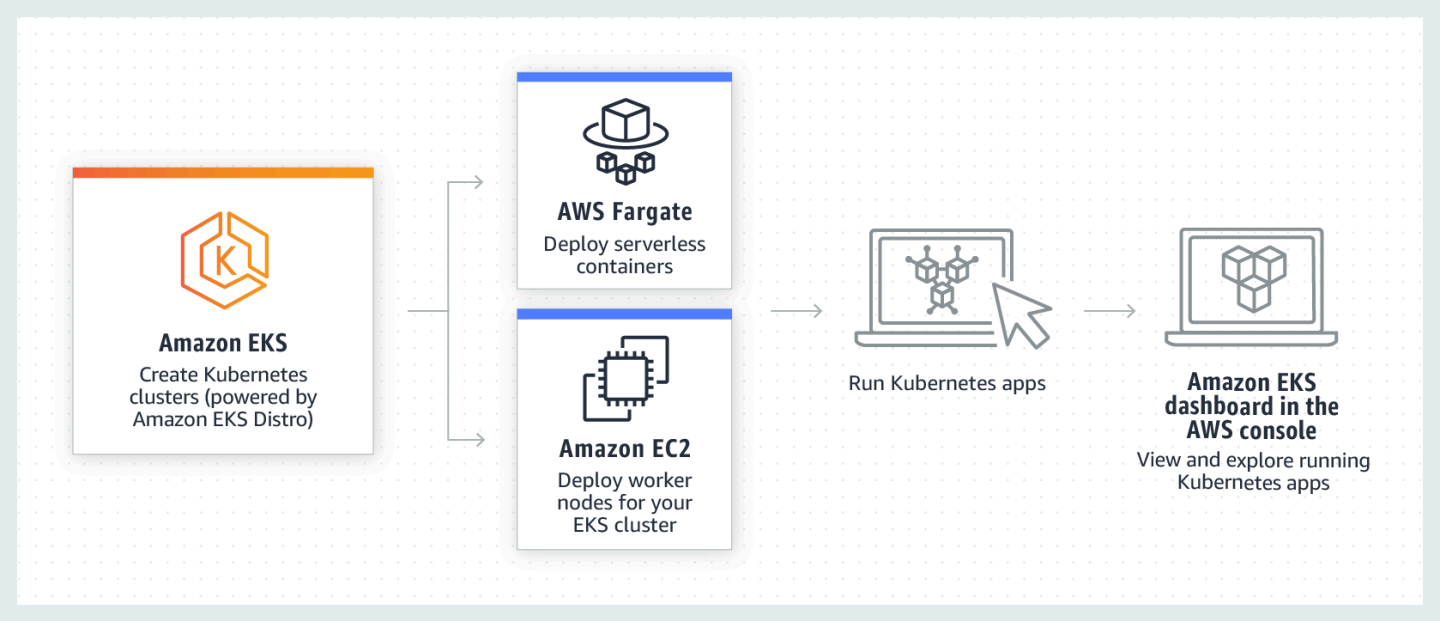
有关 Amazon EKS 其他功能的详细信息，请参阅 Amazon EC2 功能。

开始使用 Amazon EKS

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。通常，开始使用 Amazon EKS 涉及以下步骤。

1. 创建集群 - 首先使用 `eksctl`、AWS Management Console、AWS CLI 或其中一个 AWS SDK 创建集群。
2. 选择计算资源的方法 - 在 AWS Fargate、Karpenter、托管节点组和自行管理管理节点之间做出选择。
3. 设置 - 设置必要的控制器、驱动程序和服务。
4. 部署工作负载 - 量身定制您的 Kubernetes 工作负载，以充分利用所选节点类型的资源和功能。
5. 管理 - 监控您的工作负载，集成 AWS 服务以简化操作和提高工作负载性能。您可以使用 AWS Management Console 查看工作负载相关的信息。

下图显示了在云中运行 Amazon EKS 的基本流程。要了解其他 Kubernetes 部署选项，请参阅 部署选项。



Amazon EKS 的定价

Amazon EKS 集群由一个控制面板以及您在其中运行 Pods 的 Amazon Elastic Compute Cloud（Amazon EC2）或 Fargate 计算组成。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：

按需型实例

您只需按秒支付使用实例的费用，无需长期购买或预付款。有关更多信息，请参阅 Amazon EC2 按需定价和 AWS Fargate 定价。

，节省计划

您可以通过承诺在 1 年或 3 年期限内保持一致的使用量（以美元/小时为单位）来降低您的成本。有关更多信息，请参阅节省计划定价。

Amazon EKS 中的常见用例

Amazon EKS 在 AWS 上提供强大的托管 Kubernetes 服务，旨在优化容器化应用程序。以下是 Amazon EKS 的一些常见用例，可帮助您利用其优势来满足您的特定需求。

部署高可用性应用程序

借助 Elastic Load Balancing，可以确保您的应用程序在多个可用区高度可用。

构建微服务架构

将 Kubernetes 服务发现功能与 AWS Cloud Map 或 Amazon VPC Lattice 结合使用，构建弹性系统。

自动化软件发布流程

管理持续集成和持续部署（CICD）管道，简化应用程序的自动化构建、测试和部署流程。

运行无服务器应用程序

将 AWS Fargate 与 Amazon EKS 结合使用，运行无服务器应用程序。这意味着您可以只专注于应用程序开发，由 Amazon EKS 和 Fargate 来负责底层基础设施。

执行机器学习工作负载

Amazon EKS 与流行的机器学习框架兼容，比如 TensorFlow、MXNet 和 PyTorch。借助 GPU 支持，您甚至可以有效处理复杂的机器学习任务。

在本地和云端一致部署

使用 Amazon EKS Anywhere 在您自己的基础设施上运行 Kubernetes 集群，所用的工具与云中的 Amazon EKS 一致。

运行经济高效的批处理和大数据工作负载

只需很小的成本即可利用竞价型实例运行批处理和大数据工作负载，比如 Apache Hadoop 和 Spark。这让您能以折扣价利用未使用的 Amazon EC2 容量。

保护应用程序并确保合规性

实施强大的安全实践并保持对 Amazon EKS 的合规，它与 AWS Identity and Access Management（IAM）、Amazon Virtual Private Cloud（Amazon VPC）和 AWS Key Management Service (AWS KMS) 等 AWS 安全服务集成。这确保了数据隐私和保护符合行业标准。

Amazon EKS 架构

Amazon EKS 与的 Kubernetes 通用集群架构保持一致。有关更多信息，请参阅 Kubernetes 文档中的 Kubernetes 组件。以下部分介绍了 Amazon EKS 的一些额外架构细节。

控制层面

Amazon EKS 确保每个集群都有自己唯一的 Kubernetes 控制面板。这种设计使每个集群的基础设施保持独立，集群或 AWS 账户之间没有重叠。该设置包括：

分布式组件

控制面板将至少两个 API 服务器实例和三个 `etcd` 实例放置在 AWS 区域 内的三个 AWS 可用区中。

最佳性能

Amazon EKS 会主动监控和调整控制面板实例，以保持最高性能。

故障恢复能力

如果控制面板实例出现故障，Amazon EKS 会快速替换它，必要时使用不同的可用区。

稳定的正常运行时间

通过在多个可用区运行集群，实现了可靠的 API 服务器端点可用性服务水平协议（SLA）。

Amazon EKS 使用 Amazon Virtual Private Cloud（Amazon VPC）来限制单个集群中控制面板组件之间的流量。除非得到基于 Kubernetes 角色的访问控制（RBAC）策略授权，否则，集群组件无法查看或接收来自其他集群或其他 AWS 账户的通信。

计算

除了控制面板，Amazon EKS 集群还有一组称为节点的工作计算机。选择适当的 Amazon EKS 集群节点类型对于满足您的特定要求和优化资源利用率至关重要。Amazon EKS 提供以下主节点类型：

AWS Fargate

Fargate 是一个用于容器的无服务器计算引擎，无需管理底层实例。借助 Fargate，您可以指定应用程序的资源需求，然后 AWS 自动预置、扩展和维护基础设施。对于优先考虑易用性并希望专注于应用程序开发和部署而不是管理基础设施的用户来说，此选项非常适合。

Karpenter

Karpenter 是一款灵活、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 可启动适当规模的计算资源来响应不断变化的应用程序负载。此选项可以预置即时计算资源，以满足您的工作负载要求。

托管节点组

托管节点组是自动化和自定义的混合体，用于管理 Amazon EKS 集群中的 Amazon EC2 实例集合。AWS 负责修补、更新和扩展节点之类的任务，从而简化操作方面的工作。同时，还支持自定义 `kubelet` 参数，为高级 CPU 和内存管理策略提供了可能性。此外，它们还通过服务账户的 AWS Identity and Access Management（IAM）角色增强安全性，同时限制每个集群对单独权限的需求。

自行管理的节点

自行管理节点可完全控制 Amazon EKS 集群中的 Amazon EC2 实例。您负责管理、扩展和维护节点，从而完全控制底层基础设施。对于需要对其节点进行精细控制和自定义，并准备投入时间管理和维护其基础设施的用户来说，此选项非常适合。

Kubernetes 概念

Amazon Elastic Kubernetes Service（Amazon EKS）是一项基于开源 Kubernetes 项目的 AWS 托管服务。虽然您需要了解 Amazon EKS 服务如何与 AWS Cloud 集成（尤其是在您首次创建 Amazon EKS 集群时），但是一旦它启动并运行，您就可以像使用任何其他 Kubernetes 集群一样使用您的 Amazon EKS 集群。因此，要开始管理 Kubernetes 集群和部署工作负载，您至少需要对 Kubernetes 概念有基本的了解。

本页将 Kubernetes 概念分为三个部分：为什么使用 Kubernetes、集群和工作负载。第一部分描述了运行 Kubernetes 服务的价值，尤其是作为托管服务（如 Amazon EKS）运行的价值。“工作负载”部分介绍如何构建、存储、运行和管理 Kubernetes 应用程序。“集群”部分列出构成 Kubernetes 集群的不同组件，以及您在创建和维护 Kubernetes 集群方面的责任。

主题

在浏览本内容时，如果您想深入了解我们在此处介绍的任何主题，则链接将引导您进一步了解 Amazon EKS 和 Kubernetes 文档中的 Kubernetes 概念。有关 Amazon EKS 如何实现 Kubernetes 控制面板和计算功能的详细信息，请参阅 Amazon EKS 架构。

为什么选择 Kubernetes？

Kubernetes 旨在提高运行任务关键型、生产质量容器化应用程序时的可用性和可扩展性。Kubernetes 通过允许您在可以扩展或收缩以满足需求的计算机组上运行应用程序来实现这些目标，而不仅仅是在一台计算机上运行 Kubernetes（尽管可以这样做）。Kubernetes 包含能让您更轻松地执行以下操作的功能：

- 在多台机器上部署应用程序（使用部署在容器组（pod）中的容器）
- 监控容器运行状况并重启失败的容器
- 根据负载纵向扩展和缩减容器
- 使用新版本更新容器
- 在容器之间分配资源
- 平衡计算机间的流量

通过让 Kubernetes 自动执行这些类型的复杂任务，应用程序开发人员可以专注于构建和改进其应用程序工作负载，而不必担心基础设施。开发人员通常会创建格式为 YAML 文件的配置文件，这些文件用于描述应用程序的所需状态。这可能包括要运行的容器、资源限制、容器组（pod）副本数量、CPU/内存分配、关联性规则等等。

Kubernetes 的属性

为实现其目标，Kubernetes 具有以下属性：

- 容器化 - Kubernetes 是一种容器编排工具。要使用 Kubernetes，您必须先将应用程序容器化。根据应用程序的类型，它可以是一组微服务、批处理作业或其他形式。然后，您的应用程序可以利用包含庞大工具生态系统的 Kubernetes 工作流，在该工作流中，容器可以作为映像存储在容器注册表中，部署到 Kubernetes 集群，然后在可用节点上运行。在将各个容器部署到 Kubernetes 集群之前，您可以使用 Docker 或其他容器运行时在本地计算机上构建和测试这些容器。

- 可扩展 - 如果对应用程序的需求超过这些应用程序的运行实例的容量，则 Kubernetes 可以纵向扩展。根据需要，Kubernetes 可以判断应用程序是否需要更多 CPU 或内存，并通过自动扩展可用容量或使用更多的现有容量来做出响应。如果有足够的计算能力来运行更多的应用程序实例（水平容器组（pod）自动缩放），则可以在容器组（pod）级别进行扩展；如果需要调出更多节点来处理增加的容量（集群 Autoscaler 或 Karpenter），则可以在节点级别进行扩展。由于不再需要容量，这些服务可以删除不必要的容器组（pod）并关闭不需要的节点。
- 可用 - 如果应用程序或节点运行状况不佳或不可用，Kubernetes 可以将正在运行的工作负载移至另一个可用节点。您只需删除正在运行的工作负载实例或正在运行您的工作负载的节点即可强制解决问题。这里的底线是，如果工作负载不能再在当前位置运行，则可以将其调到其他位置。
- 声明式 - Kubernetes 使用主动协调来持续检查您为集群声明的状态是否与实际状态相匹配。例如，通过将 Kubernetes 对象应用于集群（通常通过 YAML 格式的配置文件），您可以要求启动要在集群上运行的工作负载。您可以稍后更改配置，以执行诸如使用更高版本的容器或分配更多内存之类的操作。Kubernetes 将通过完成需要的操作来建立所需的状态。这可能包括启动或关闭节点、停止和重启工作负载，或拉取更新的容器。
- 可组合 - 由于应用程序通常由多个组件组成，因此您希望能够同时管理其中一组组件（通常由多个容器表示）。虽然 Docker Compose 提供使用 Docker 直接执行此操作的方法，但 Kubernetes Kompose 命令可以帮助您使用 Kubernetes 执行该操作。有关如何执行此操作的示例，请参阅将 Docker Compose 文件翻译成 Kubernetes 资源。
- 可扩展 - 与专有软件不同的是，开源 Kubernetes 项目旨在向您开放，从而以您喜欢的任何方式来扩展 Kubernetes 以满足您的需求。API 和配置文件可以直接修改。鼓励第三方编写自己的控制器，以同时扩展基础设施和最终用户 Kubernetes 功能。Webhook 使您能够设置集群规则，以强制执行策略并适应不断变化的条件。有关如何扩展 Kubernetes 集群的更多想法，请参阅扩展 Kubernetes。
- 可移植 - 许多组织已将其在 Kubernetes 上的操作标准化，因为它允许他们以相同的方式管理所有应用程序需求。开发人员可以使用相同的管道来构建并存储容器化应用程序。然后，可以将这些应用程序部署到在本地、云、餐厅的销售点终端上，或分散在公司远程站点的 IOT 设备上运行的 Kubernetes 集群中。它的开源性质使人们有可能开发这些特殊的 Kubernetes 发行版，以及管理它们所需的工具。

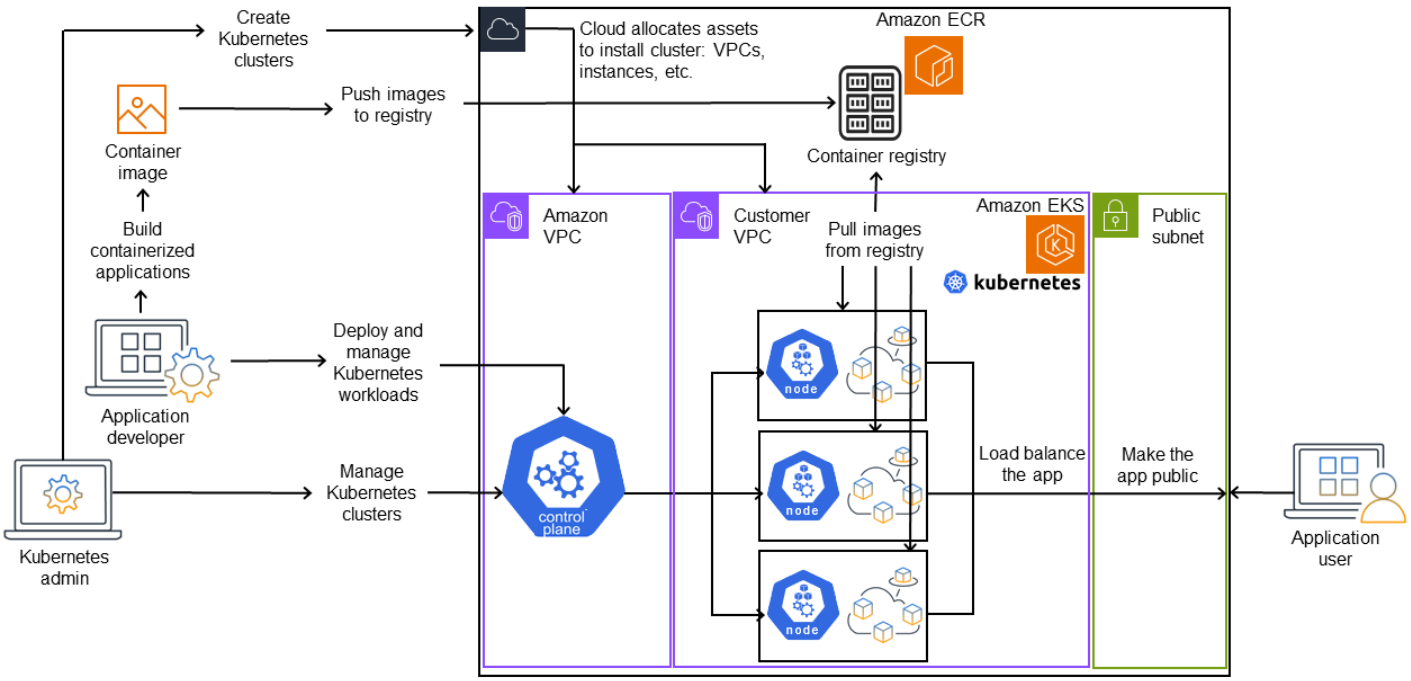
管理 Kubernetes

Kubernetes 源代码免费提供，因此您可以使用自己的设备自行安装和管理 Kubernetes。但是，自行管理 Kubernetes 需要深厚的运营专业知识，并且需要时间和精力来维护。出于这些原因，大多数部署生产工作负载的人员都会选择云提供商（例如 Amazon EKS）或本地提供商（例如 Amazon EKS Anywhere），这些提供商拥有经过测试的 Kubernetes 分配和 Kubernetes 专家支持。这使您可以卸下维护集群所需的大量千篇一律的工作，包括：

- 硬件 - 如果您没有可根据要求运行 Kubernetes 的硬件，则 AWS Amazon EKS 等云提供商可以为您节省前期成本。借助 Amazon EKS，这意味着您可以使用 AWS 提供的最佳云资源，包括计算机实例（Amazon Elastic Compute Cloud）、您自己的私有环境（Amazon VPC）、中央身份和权限管理（IAM）和存储（Amazon EBS）。AWS 管理计算机、网络、数据中心和运行 Kubernetes 所需的所有其他物理组件。同样，您不必计划数据中心在需求最高的时期处理最大容量。对于 Amazon EKS Anywhere 或其他本地 Kubernetes 集群，您负责管理 Kubernetes 部署中使用的基础设施，但您仍然可以依靠 AWS 来帮助您保持 Kubernetes 最新状态。
- 控制面板管理 - Amazon EKS 管理 AWS 托管的 Kubernetes 控制面板的安全性和可用性，控制面板负责计划容器、管理应用程序的可用性以及其他关键任务，因此您可以专注于应用程序工作负载。如果您的集群中断，则 AWS 应该有办法将集群恢复到运行状态。对于 Amazon EKS Anywhere，您将自己管理控制面板。
- 经过测试的升级 - 升级集群时，您可以依靠 Amazon EKS 或 Amazon EKS Anywhere 来提供其 Kubernetes 分配的测试版本。
- 插件 - 有数百个项目为扩展和使用 Kubernetes 而构建，您可以将其添加到集群的基础设施中，也可以用它们来帮助工作负载的运行。AWS 提供可用于集群的 Amazon EKS 插件，而不是自己构建和管理这些插件。Amazon EKS Anywhere 提供精选软件包，其中包括许多热门开源项目的版本。因此，您不必自己构建软件，也不必管理关键的安全补丁、漏洞修复或升级。同样，如果默认设置满足您的需求，则通常只需要对这些插件进行很少的配置。有关使用插件扩展集群的详细信息，请参阅扩展集群。

操作中的 Kubernetes

下图显示您作为 Kubernetes 管理员或应用程序开发人员在创建和使用 Kubernetes 集群时要执行的关键活动。在此过程中，它将 AWS 云作为底层云提供商的示例，说明了 Kubernetes 组件是如何相互交互的。



Kubernetes 管理员使用特定于将在其上构建集群的提供商类型的工具来创建 Kubernetes 集群。此示例使用 AWS 云作为提供商，它提供名为 Amazon EKS 的托管 Kubernetes 服务。托管服务会自动分配创建集群所需的资源，包括为集群创建两个新的 Virtual Private Cloud（Amazon VPC）、设置网络、将 Kubernetes 权限映射到管理云中资产的权限、查看控制面板服务的运行位置，以及分配零个或多个 Amazon EC2 实例作为运行工作负载的 Kubernetes 节点。AWS 为控制面板自行管理一个 Amazon VPC，而另一个 Amazon VPC 则包含运行工作负载的客户节点。

Kubernetes 管理员今后的许多任务都是使用 kubectl 之类的 Kubernetes 工具完成的。该工具直接向集群的控制面板发出服务请求。因此，对集群进行查询和更改的方式与您在任何 Kubernetes 集群上执行查询和更改的方式非常相似。

想要将工作负载部署到此集群的应用程序开发人员可以执行多项任务。开发人员需要将应用程序构建到一个或多个容器映像中，然后将这些映像推送到 Kubernetes 集群可以访问的容器注册表。AWS 为此提供 Amazon Elastic Container Registry（Amazon ECR）。

要运行应用程序，开发人员可以创建 YAML 格式的配置文件，告诉集群如何运行应用程序，包括要从注册表中提取哪些容器以及如何将这些容器封装在容器组（pod）中。控制面板（调度程序）将容器调度到一个或多个节点，每个节点上的容器运行时实际上会拉取并运行所需的容器。开发人员还可以设置应用程序负载均衡器，以均衡在每个节点上运行的可用容器的流量，并公开应用程序，使其可以在公共网络上对外开放。完成所有操作后，想要使用应用程序的人可以连接到应用程序端点进行访问。

以下部分将从 Kubernetes 集群和工作负载的角度详细介绍每项功能。

集群

如果您的作业是启动和管理 Kubernetes 集群，则应了解 Kubernetes 集群是如何创建、增强、管理和删除的。您还应该了解构成集群的组件是什么，以及需要做些什么来维护这些组件。

用于管理集群的工具可以处理 Kubernetes 服务与底层硬件提供商之间的重叠问题。因此，这些任务的自动化往往由 Kubernetes 提供商（例如 Amazon EKS 或 Amazon EKS Anywhere）使用特定于提供商的工具来完成。例如，要启动 Amazon EKS 集群，您可以使用 `eksctl create cluster`，而对于 Amazon EKS Anywhere，您可以使用 `eksctl anywhere create cluster`。请注意，虽然这些命令创建 Kubernetes 集群，但它们是特定于提供商的，而不是 Kubernetes 项目本身的一部分。

集群创建和管理工具

Kubernetes 项目提供手动创建 Kubernetes 集群的工具。因此，如果您想在单台计算机上安装 Kubernetes，或者在计算机上运行控制面板并手动添加节点，则可以使用 Kubernetes 安装工具下列出的 kind、minikube 或 kubeadm 之类的 CLI 工具。要简化和自动执行集群创建和管理的整个生命周期，使用现有 Kubernetes 提供商（例如 Amazon EKS 或 Amazon EKS Anywhere）支持的工具要容易得多。

在 AWS Cloud 中，您可以使用 CLI 工具（例如 eksctl）或更多声明性工具（例如 Terraform）创建 Amazon EKS 集群（请参阅 Terraform 的 Amazon EKS 蓝图）。您也可以从 AWS 管理控制台创建集群。请参阅 Amazon EKS 功能，查看您使用 Amazon EKS 获得的列表。Amazon EKS 为您承担的 Kubernetes 责任包括：

- 托管控制面板 - AWS 确保 Amazon EKS 集群可用且可扩展，因为它为您管理控制面板并使该面板在 AWS 可用区域中可用。
- 节点管理 - 您可以让 Amazon EKS 根据需要使用托管节点组或 Karpenter 自动创建节点，而不是手动添加节点。托管节点组已与 Kubernetes 集群自动扩展集成。使用节点管理工具，您可以利用节省的成本与竞价型实例和节点整合，以及可用性等项目，使用计划功能来设置工作负载的部署方式和节点的选择方式。
- 集群联网 - 使用 CloudFormation 模板，`eksctl` 在 Kubernetes 集群中的控制面板和数据面板（节点）组件之间设置联网。它还设置可以通过其进行内部和外部通信的端点。有关详细信息，请参阅揭秘 Amazon EKS 工作线程节点的集群联网。Amazon EKS 中的容器组（pod）之间的通信使用 Amazon EKS 容器组身份完成，这提供了一种让容器组（pod）利用 AWS 云方法来管理凭证和权限的方式。
- 插件 - Amazon EKS 使您不必构建和添加通常用于支持 Kubernetes 集群的软件组件。例如，当您从 AWS 管理控制台创建 Amazon EKS 集群时，它会为 Kubernetes 和 CoreDNS 附加组件添加

Amazon EKS kube-proxy、Amazon VPC CNI 插件。有关这些附加组件的更多信息，包括可用附件组件列表，请参阅 Amazon EKS 附加组件。

为了在您自己的本地计算机和网络上运行集群，Amazon 提供了 Amazon EKS Anywhere。您可以选择使用自己的设备在 VMWare vSphere、裸机（Tinkerbell 提供程序）、Snow、CloudStack 或 Nutanix 平台上运行 Amazon EKS Anywhere，而不是将 AWS Cloud 作为提供商。

Amazon EKS Anywhere 基于 Amazon EKS 使用的相同 Amazon EKS Distro 软件。但是，Amazon EKS Anywhere 依赖 Kubernetes 集群 API（CAPI）接口的不同实现来管理 Amazon EKS Anywhere 集群中计算机的整个生命周期（例如 vSphere 的 CAPV 和 CloudStack 的 CAPC）。由于整个集群都在您的设备上运行，因此您还要承担管理控制面板和备份其数据的额外责任（请参阅本文档后面的 etcd）。

集群组件

Kubernetes 集群组件分为两个主要区域：控制面板和 Worker 节点。控制面板组件管理集群并提供对其 API 的访问权限。Worker 节点（有时简称为“节点”）提供运行实际工作负载的地方。节点组件由在每个节点上运行的服务组成，这些服务用于与控制面板通信并运行容器。您的集群的 Worker 节点组称为数据面板。

控制层面

控制面板由一组管理集群的服务组成。这些服务可能全部在单台计算机上运行，也可能分布在多台计算机上。在内部，它们被称为控制面板实例（CPI）。CPI 的运行方式取决于集群的大小和对高可用性的要求。随着集群中的需求增加，控制面板服务可以扩展以提供该服务的更多实例，并在实例之间实现请求的负载均衡。

Kubernetes 控制面板组件执行的任务包括：

- 与集群组件（API 服务器）通信 - API 服务器（kube-apiserver）公开 Kubernetes API，因此对集群的请求可以从集群内部和外部发出。换句话说，添加或更改集群对象（容器组（pod）、服务、节点等）的请求可能来自外部命令，例如来自 `kubectl` 的运行容器组（pod）的请求。同样，可以从 API 服务器向集群内的组件发出请求，例如向 `kubelet` 服务查询容器组（pod）的状态。
- 存储有关集群的数据（etcd 键值存储）- etcd 服务提供跟踪集群当前状态的关键作用。如果 etcd 服务变得不可访问，则您将无法更新或查询集群的状态，尽管工作负载会继续运行一段时间。因此，关键集群通常会同时运行多个负载均衡的 etcd 服务实例，并定期备份 etcd 键值存储以防数据丢失或损坏。请记住，在 Amazon EKS 中，这一切都是默认自动为您处理的。Amazon EKS Anywhere 提供 etcd 备份和恢复的说明。请参阅 etcd 数据模型，了解 etcd 管理数据的方式。

- 将容器组（pod）调度到节点（调度程序）- 启动或停止 Kubernetes 中的容器组（pod）的请求会被定向到 Kubernetes 调度程序（kube-scheduler）。由于一个集群可能有多个能够运行容器组（pod）的节点，因此应由调度程序来选择容器组（pod）应在哪个节点（或哪些节点，如果是副本）上运行。如果没有足够的可用容量在现有节点上运行所请求的容器组（pod），除非您进行了其他规定，否则请求将失败。这些规定可能包括启用托管节点组或 Karpenter 之类的服务，这些服务可以自动启动新节点来处理工作负载。
- 将组件保持在所需状态（控制器管理器）- Kubernetes 控制器管理器作为进程守护程序（kube-controller-manager）运行，以监视集群的状态并对集群进行更改以重新建立预期状态。特别是，有几个控制器监视不同的 Kubernetes 对象，其中包括 node-lifecycle-controller、statefulset-controller、endpoint-controller、cronjob-controller 等。
- 管理云资源（云控制器管理器）- Kubernetes 与执行底层数据中心资源的请求的云提供商之间的交互由云控制器管理器（cloud-controller-manager）处理。由云控制器管理器管理的控制器可以包括路由控制器（用于设置云网络路由）、服务控制器（用于使用云负载均衡服务）和节点控制器（用于使用云 API 让 Kubernetes 节点与云节点保持同步）。

Worker 节点（数据面板）

对于单节点 Kubernetes 集群，工作负载与控制面板在同一台计算机上运行。但是，更普通的配置是拥有一个或多个专门用于运行 Kubernetes 工作负载的独立计算机系统（节点）。

首次创建 Kubernetes 集群时，您可以通过某些集群创建工具配置一定数量的节点以添加到集群中（通过识别现有的计算机系统或让提供商创建新的计算机系统）。在向这些系统添加任何工作负载之前，向每个节点添加服务以实现以下功能：

- 管理每个节点（kubelet）- API 服务器与每个节点上运行的 kubelet 服务进行通信，以确保节点已正确注册且调度程序请求的容器组（pod）正在运行。kubelet 可以读取容器组（pod）清单，并在本地系统上设置容器组（pod）所需的存储卷或其他功能。它还可以检查本地运行的容器的运行状况。
- 在节点上运行容器（容器运行时）- 每个节点上的容器运行时管理分配给节点的每个容器组（pod）所请求的容器。这意味着它可以从相应的注册表中拉取容器映像、运行容器、停止容器，并响应有关容器的查询。默认的容器运行时是 containerd。截至 Kubernetes 1.24，可以用作容器运行时的 Docker（Dockershim）的特殊集成已从 Kubernetes 中删除。虽然您仍然可以使用 Docker 在本地系统上测试和运行容器，但要将 Docker 与 Kubernetes 结合使用，您现在必须在每个节点上安装 Docker 引擎才能将其与 Kubernetes 结合使用。
- 管理容器之间的联网（kube-proxy）- 为了能够使用服务支持容器组（pod）之间的通信，Kubernetes 需要一种方法来设置容器组（pod）网络来跟踪与这些容器组（pod）关联的 IP 地址和端口。kube-proxy 服务在每个节点上运行，以允许容器组（pod）之间进行通信。

扩展集群

您可以添加一些服务到 Kubernetes 中来支持集群，但不能在控制面板中运行这些服务。这些服务通常直接在 kube-system 命名空间或其自己的命名空间中的节点上运行（就像第三方服务提供商经常做的那样）。一个常见的例子是 CoreDNS 服务，它为集群提供 DNS 服务。有关如何查看集群上的 kube-system 中正在运行哪些集群服务的信息，请参阅发现内置服务。

您可以考虑向集群添加不同类型的附加组件。为了保持集群正常运行，您可以添加使您能够执行日志记录、审计等操作的可观测性功能，以及指标。使用这些信息，您就可以对发生的问题进行排查，通常通过相同的可观测性接口排查。这些类型的服务的示例包括 Amazon GuardDuty、CloudWatch、AWS Distro for OpenTelemetry、Kubernetes 的 Amazon VPC CNI 插件和 Grafana Kubernetes 监控。对于存储，Amazon EKS 的插件包括 Amazon Elastic Block Store CSI 驱动程序（用于添加块存储设备）、Amazon Elastic File System CSI 驱动程序（用于添加文件系统存储）和多个第三方存储插件（例如适用于 NetApp ONTAP 的 Amazon FSx CSI 驱动程序）。

有关可用的 Amazon EKS 插件的更完整列表，请参阅 Amazon EKS 插件。

工作负载

Kubernetes 将工作负载定义为“在 Kubernetes 上运行的应用程序”。该应用程序可以由一组在容器组（pod）中作为容器运行的微服务组成，也可以作为批处理作业或其他类型的应用程序运行。Kubernetes 的作业是确保您为要设置或部署的对象发出的请求得到执行。作为部署应用程序的人，您应该了解容器是如何构建的、容器组（pod）是如何定义的，以及您可以使用哪些方法来部署它们。

容器

您在 Kubernetes 中部署和管理的应用程序工作负载中最基本的元素是容器组（pod）。容器组（pod）代表一种保存应用程序组件以及定义描述容器组（pod）属性的规范的方法。与此形成鲜明对比的是 RPM 或 Deb 软件包，后者将适用于 Linux 系统的软件打包在一起，但其本身并不作为一个实体运行。

由于容器组（pod）是最小的可部署单元，因此它通常只能容纳一个容器。但是，如果容器紧密耦合，则一个容器组（pod）中可以有多个容器。例如，Web 服务器容器可能被打包在带有 sidecar 类型的容器的容器组（pod）中，该容器类型可以提供日志记录、监控或其他与 Web 服务器容器紧密相关的服务。在这种情况下，位于同一容器组（pod）中可确保对于容器组（pod）的每个正在运行的实例，两个容器始终在同一个节点上运行。同样，容器组（pod）中的所有容器共享同一个环境，容器组（pod）中的容器就像在同一个隔离主机中一样运行。这样做的效果是，容器共享一个提供对容器组（pod）的访问权限的 IP 地址，并且容器可以像在自己的本地主机上运行一样相互通信。

容器组（pod）规范（PodSpec）定义容器组（pod）的所需状态。您可以通过使用工作负载资源管理容器组（pod）模板来部署单个或多个容器组（pod）。工作负载资源包括部署（用于管理多个容器组（pod）副本）、StatefulSets（用于部署需要具有唯一性的容器组（pod），例如数据库容器组（pod））和 DaemonSets（其中容器组（pod）需要在每个节点上持续运行）。稍后会详细介绍。

虽然容器组（pod）是您部署的最小单元，但容器是您构建和管理的最小单元。

构建容器

容器组（pod）实际上只是围绕一个或多个容器的结构，每个容器本身都包含文件系统、可执行文件、配置文件、库和其他用于实际运行应用程序的组件。由于一家名为 Docker Inc. 的公司首先推广了容器，因此有些人将容器称为 Docker 容器。但是，开放容器计划此后为该行业定义了容器运行时、映像和分发方法。再加上容器是由许多现有的 Linux 功能创建的这一事实，另一些人通常将容器称为 OCI 容器、Linux 容器或仅仅是容器。

当您构建容器时，不变通常会从一个 Docker 文件（字面上的意思）开始。在那个 Dockerfile 中，您可以识别：

- 基础映像 - 基础容器映像是一种容器，通常由操作系统文件系统的最小版本（例如 Red Hat Enterprise Linux 或 Ubuntu）或经过增强以提供运行特定类型应用程序（例如 nodejs 或 python 应用程序）的软件的最小系统构建。
- 应用程序软件 - 您可以将应用程序软件添加到容器中，方法与将其添加到 Linux 系统中的方法大致相同。例如，在您的 Dockerfile 中，您可以运行 npm 和 `yarn` 安装 Java 应用程序，或运行 yum 和 `dnf` 安装 RPM 软件包。换句话说，在 Dockerfile 中使用 RUN 命令，您可以运行基础映像文件系统中提供的任何命令，以在生成的容器映像中安装软件或配置软件。
- 指令 - Dockerfile 参考描述了在配置 Dockerfile 时可以将其添加到 Dockerfile 中的指令。这些指令包括用于构建容器本身（本地系统的 `ADD` 或 `COPY` 文件）中的内容、识别容器运行时要执行的命令（`CMD` 或 `ENTRYPOINT`），以及将容器连接到其运行的系统（通过识别要运行的身份 `USER`、要挂载的本地 `VOLUME` 或要 `EXPOSE` 的端口）的指令。

虽然传统上使用 `docker` 命令和服务来构建容器（`docker build`，但其他可用于构建容器映像的工具包括 podman 和 nerdctl。要了解如何构建容器，请参阅构建更好的容器映像或使用 Docker 构建。

存储容器

构建容器镜像后，可以将其存储在工作站上的容器分发注册表或公共容器注册表中。在工作站上运行私有容器注册表可使您在本地存储容器映像，使它们随时可供您使用。

要以更公开的方式存储容器映像，可以将其推送到公共容器注册表。公共容器注册表为存储和分发容器映像提供了一个中心位置。公共容器注册表的示例包括 Amazon Elastic Container Registry、Red Hat Quay 注册表和 Docker Hub 注册表。

在 Amazon Elastic Kubernetes Service（Amazon EKS）上运行容器化工作负载时，建议您拉取存储在 Amazon Elastic Container Registry 中的 Docker 官方映像的副本。AWS自 2021 年以来，Amazon ECR 一直在存储这些映像。您可以在 Amazon ECR 公开映像浏览馆中搜索常用容器映像，特别是对于 Docker Hub 映像，您可以搜索 Amazon ECR Docker 库。

正在运行的容器

由于容器是以标准格式构建的，因此容器可以在任何可运行容器运行时（例如 Docker）且其内容与本地计算机的架构（例如 x86_64 或 arm）相匹配的计算机上运行。要测试容器或者只是在本地桌面上运行它，您可以使用 `docker run` 或 `podman run` 命令在本地主机上启动容器。但是对于 Kubernetes，每个 Worker 节点都部署了容器运行时，它可以依据 Kubernetes 请求节点运行容器。

将容器分配到某个节点上运行后，该节点将查看节点上是否已存在所请求的容器映像版本。如果没有，则 Kubernetes 告诉容器运行时从相应的容器注册表中拉取该容器，然后在本地运行该容器。请记住，容器映像是指在笔记本电脑、容器注册表和 Kubernetes 节点之间移动的软件包。容器是指该映像的运行实例。

容器组（pod）

容器准备就绪后，使用容器组（pod）的操作包括配置、部署和使容器组（pod）可访问。

配置容器组（pod）

当您定义容器组（pod）时，会为其分配一组属性。这些属性必须至少包含容器组（pod）名称和要运行的容器映像。但是，您还需要使用容器组（pod）定义配置许多其他内容（有关可以进入容器组（pod）的内容的详细信息，请参阅 PodSpec 页面）。其中包括：

- 存储 - 停止并删除正在运行的容器后，该容器中的数据存储将消失，除非您设置更永久的存储。Kubernetes 支持许多不同的存储类型，并在卷的保护下对它们进行抽象。存储类型包括 CephFS、NFS、iSCSI 等。您甚至可以在本地计算机上使用本地块设备。使用集群中提供的其中一种存储类型，您可以将存储卷挂载到容器文件系统中的选定挂载点。永久卷是指删除容器组（pod）后继续存在的卷，而临时卷在容器组（pod）被删除时会被删除。如果您的集群管理员为您的集群创建了不同的存储类，则可以选择所用存储的属性，例如卷在使用后是删除还是回收，如果需要更多空间，则它是否会扩展，甚至它是否满足某些性能要求。
- 密钥 - 通过在容器组（pod）规范中向容器提供密钥，您可以提供这些容器访问文件系统、数据库或其他受保护资产所需的权限。密钥、密码和令牌是可以作为密钥存储的项目之一。使用密钥使您

不必将这些信息存储在容器映像中，而只需要将密钥提供给正在运行的容器即可。与密钥相似的是 ConfigMaps。ConfigMap 往往会保存不太重要的信息，例如用于配置服务的键值对。

- 容器资源 - 用于进一步配置容器的对象可以采用资源配置的形式。对于每个容器，您可以请求其可以使用的内存和 CPU 量，以及对容器可以使用的资源总量设置限制。有关示例，请参阅容器组（pod）和容器的资源管理。
- 中断 - 容器组（pod）可以被非自愿中断（节点关闭），也可以自愿中断（需要升级）。通过配置容器组（pod）中断预算，您可以在一定程度上控制发生中断时应用程序的可用性。有关示例，请参阅为您的应用程序指定中断预算。
- 命名空间 - Kubernetes 提供不同的方法来相互隔离 Kubernetes 组件和工作负载。在同一个命名空间中运行特定应用程序的所有容器组（pod）是共同保护和管理这些容器组（pod）的常用方法。您可以创建自己的命名空间以供使用，也可以选择不指定命名空间（这会导致 Kubernetes 使用 `default` 命名空间）。Kubernetes 控制面板组件通常在 kube-system 命名空间中运行。

刚才描述的配置通常汇集在要应用于 Kubernetes 集群的 YAML 文件中。对于个人 Kubernetes 集群，您可以将这些 YAML 文件存储在本地系统上。但是，对于更关键的集群和工作负载，GitOps 是自动存储和更新工作负载和 Kubernetes 基础设施资源的常用方式。

用于将容器组（pod）信息收集在一起并进行部署的对象由以下部署方法之一定义。

部署 Pod

您为部署容器组（pod）而选择的方法取决于您计划使用这些容器组（pod）运行的应用程序的类型。以下是您的一些选择：

- 无状态应用程序 - 无状态应用程序不保存客户端的会话数据，因此另一个会话无需重新引用前一个会话发生的情况。这使得在容器组（pod）变得运行不正常时将其替换为新的容器组（pod）或者在不保存状态的情况下移动它们会更容易。如果您运行的是无状态应用程序（例如 Web 服务器），则可以使用部署来部署容器组（pod）和 ReplicaSets。ReplicaSet 定义您想同时运行容器组（pod）的多少个实例。尽管您可以直接运行 ReplicaSet，但通常是在部署中直接运行副本，以定义一个容器组（pod）一次应该运行多少个副本。
- 有状态应用程序 - 有状态应用程序是看中容器组（pod）的身份和容器组（pod）的启动顺序的应用程序。这些应用程序需要稳定的永久存储，并且需要以一致的方式进行部署和横向缩减。要在 Kubernetes 中部署有状态应用程序，可以使用 StatefulSets。通常以 StatefulSet 形式运行的应用程序的一个示例是数据库。在 StatefulSet 中，您可以定义副本、容器组（pod）及其容器、要挂载的存储卷以及容器中存储数据的位置。有关将数据库部署为 ReplicaSet 的示例，请参阅运行已复制的有状态应用程序。

- 每节点应用程序 - 有时您想在 Kubernetes 集群中的每个节点上运行应用程序。例如，您的数据中心可能要求每台计算机都运行监控应用程序或特定的远程访问服务。对于 Kubernetes，您可以使用 DaemonSet 确保所选应用程序在集群中的每个节点上运行。
- 应用程序运行至完成 - 您希望运行一些应用程序来完成特定的任务。这可能包括运行月度状态报告或清理旧数据的应用程序。Job 对象可用于设置应用程序以启动和运行，然后在任务完成后退出。CronJob 对象允许您使用由 Linux crontab 格式定义的结构将应用程序设置为在特定的小时、分钟、一月中的某一天、月份或一周中的某一天运行。

使应用程序可以从网络访问

由于应用程序通常部署为一组会四处移动到不同地方的微服务，因此 Kubernetes 需要一种方法让这些微服务能够相互找到。此外，为了让其他人访问 Kubernetes 集群之外的应用程序，Kubernetes 需要一种在外部地址和端口上公开该应用程序的方法。这些与联网相关的功能分别通过 Service 和 Ingress 对象完成：

- 服务 - 由于容器组（pod）可以移动到不同的节点和地址，因此需要与第一个容器组（pod）通信的另一个容器组（pod）可能会发现很难找到它的位置。要解决此问题，Kubernetes 可以让您将应用程序表示为服务。使用服务，您可以识别具有特定名称的一个容器组（pod）或一组容器组（pod），然后指明哪个端口从容器组（pod）中公开该应用程序的服务，以及其他应用程序可以使用哪些端口来联系该服务。集群中的另一个容器组（pod）只需按名称请求服务，然后 Kubernetes 将该请求定向到运行该服务的容器组（pod）实例的正确端口。
- Ingress - Ingress 可以使以 Kubernetes 服务为代表的应用程序可供集群外的客户端使用。Ingress 的基本功能包括负载均衡器（由 Ingress 管理）、Ingress 控制器以及将请求从控制器路由到服务的规则。有几个 Ingress 控制器可供您使用 Kubernetes 选择。

后续步骤

了解基本 Kubernetes 概念及其与 Amazon EKS 的关系将有助于您浏览 Amazon EKS 文档和 Kubernetes 文档，以找到您管理 Amazon EKS 集群和向这些集群部署工作负载所需的信息。要开始使用 Amazon EKS，请从以下选项中进行选择：

- 创建简单集群
- 创建更复杂的集群
- 部署示例应用程序
- 探索管理集群的方法

部署选项

您可以使用以下任一选项部署 Amazon EKS：

云中的 Amazon EKS

您可以在 AWS 云中运行 Kubernetes，而无需安装、操作和维护自己的 Kubernetes 控制面板或节点。本指南将介绍此选项。

Amazon EKS on Outposts

AWS Outposts 在本地设施中启用本机 AWS 服务、基础设施和操作模型。借助 Amazon on Outposts，您可以选择运行扩展集群或本地集群。借助扩展集群，Kubernetes 控制面板在 AWS 区域 中运行，而节点在 Outposts 上运行。借助本地集群，整个 Kubernetes 集群在 Outpost 上本地运行，包括 Kubernetes 控制面板和节点。有关更多信息，请参阅AWS Outposts 上的 Amazon EKS。

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。有关一些常见问题的答案，请参阅 Amazon EKS Anywhere 常见问题。

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 EKS on Outposts	Amazon EKS Anywhere	Amazon EKS Distro
硬件	AWS 提供	AWS 提供	由您提供	由您提供
部署位置	AWS 云	您的数据中心	您的数据中心	您的数据中心

功能	Amazon EKS	Amazon EKS on Outposts	Amazon EKS Anywhere	Amazon EKS Distro
Kubernetes 控制面板位置	AWS 云	AWS 云或您的数据中心	您的数据中心	您的数据中心
Kubernetes 数据面板位置	AWS 云	您的数据中心	您的数据中心	您的数据中心
支持	AWS Support	AWS Support	AWS Support	OSS 社群支持

设置使用 Amazon EKS

AWS 资源通常具有访问限制，用于限制对创建这些资源的 AWS 实体的访问。因此，从一开始就在 AWS Command Line Interface 中建立正确的用户配置至关重要。此外，您还需要为本地计算机配备必要的工具，以便通过命令行高效地管理 Amazon EKS 集群。本主题将帮助您为集群的命令行管理做好准备。

步骤 1：设置 AWS CLI

AWS CLI 是与 AWS 服务一起使用的命令行工具，包括 Amazon EKS。该工具还可用于对从本地计算机访问 Amazon EKS 集群和其他 AWS 资源的 IAM 用户或角色进行身份验证。要通过命令行在 AWS 中预置资源，您需要获取 AWS 访问密钥 ID 和私有密钥，以便在命令行中使用。然后，您需要在 AWS CLI 中配置这些凭证。如果尚未安装 AWS CLI，请参阅《AWS Command Line Interface 用户指南》中的安装或更新 AWS CLI 的最新版本。

创建访问密钥

1. 登录到 AWS Management Console。
2. 在右上角，选择您的 AWS 用户名以打开导航菜单。例如，选择 **webadmin**。然后选择安全凭证。
3. 在访问密钥下，选择创建访问密钥。
4. 选择命令行界面（CLI），然后选择下一步。
5. 选择创建访问密钥。
6. 选择下载 .csv 文件。

配置AWS CLI

安装 AWS CLI 后，请按照以下步骤进行配置。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI。

1. 在终端窗口中，输入以下命令：

   ```
   aws configure
   ```

   您还可以选择配置一个命名配置文件，例如 **--profile cluster-admin**。如果您在 AWS CLI 中配置了命名配置文件，则必须始终在后续命令中传递此标志。

2. 输入 AWS 凭证。例如：

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: region-code
Default output format [None]: json
```

获取安全令牌

如果需要，运行以下命令以获取 AWS CLI 的新安全令牌。有关更多信息，请参阅《AWS CLI 命令参考》中的 get-session-token。

默认情况下，令牌的有效期为 15 分钟。要更改默认会话超时，请传递 **--duration-seconds** 标志。例如：

```
aws sts get-session-token --duration-seconds 3600
```

此命令将返回 AWS CLI 会话的临时安全凭证。您应看到以下响应输出：

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

验证用户身份

如果需要，可运行以下命令来验证用于终端会话的 IAM 用户身份（例如 *ClusterAdmin*）的 AWS 凭证。

```
aws sts get-caller-identity
```

此命令将返回为 AWS CLI 配置的 IAM 实体的 Amazon 资源名称（ARN）。您应看到以下示例响应输出：

```
{
```

```
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
}
```

步骤 2：安装 Kubernetes 工具

要与 Kubernetes 集群通信，您需要一个与 Kubernetes API 进行交互的工具。此外，您还需要一些其他工具，例如用于管理本地计算机上 Kubernetes 环境的工具。

创建 AWS 资源

- Amazon EKS 集群资源 – 如果您不熟悉 AWS，我们建议您安装 `eksctl`。`eksctl` 是一个基础设施即代码（IaC）实用程序，该实用程序可使用 AWS CloudFormation 来轻松创建 Amazon EKS 集群。该实用程序还会创建其他 Kubernetes 资源，例如服务账户。有关安装 `eksctl` 的说明，请参阅 `eksctl` 文档中的 Installation。
- AWS 资源 – 如果您习惯于自动预置和部署 AWS 基础设施，我们建议您安装 Terraform。Terraform 是由 HashiCorp 开发开源基础设施即代码（IaC）工具。该工具允许您使用高级配置语言 [例如 HashiCorp 配置语言（HCL）或 JSON] 来定义和预置基础设施。有关安装 Terraform 的说明，请参阅 Terraform 文档中的 Install Terraform。

要安装 **kubectl**，请执行以下操作

`kubectl` 是一款开源命令行工具，用于与 Amazon EKS 集群上的 Kubernetes API 服务器进行通信。如果还没有在本地计算机上安装，请从以下选项中选择。

- AWS 版本 – 要安装 Amazon EKS 支持的 `kubectl` 版本，请参阅 安装或更新 `kubectl`。
- 社区版本 – 要安装 `kubectl` 的最新社区版本，请参阅 Kubernetes 文档中的安装工具页面。

设置开发环境

- 本地部署工具 – 如果您不熟悉 Kubernetes，可以考虑安装像 `minikube` 或 `kind` 这样的本地部署工具。这些工具允许您在本地计算机上管理 Amazon EKS 集群。
- 软件包管理器 – Helm 是 Kubernetes 一款常用的软件包管理器，该工具简化了复杂软件包的安装和管理。通过 Helm，可以更轻松地在 Amazon EKS 集群上安装和管理诸如 AWS 负载均衡器控制器之类的软件包。

后续步骤

- 开始使用 Amazon EKS

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 AWS 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，1.29 kubectl 客户端使用 Kubernetes 1.28、1.29 和 1.30 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 --client
   ```

   如果您已将 kubectl 安装在设备路径中，示例输出包括类似于如下的信息。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。

   ```
   Client Version: v1.30.X-eks-1234567
   ```

   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 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/darwin/amd64/kubectl
```

- Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/darwin/amd64/kubectl
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/darwin/amd64/kubectl
```

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

   a. 下载集群 Kubernetes 版本的 SHA-256 校验和。

   - Kubernetes 1.30

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.29

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.28

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.27

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.26

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.25

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/darwin/amd64/kubectl.sha256
   ```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/darwin/amd64/kubectl.sha256
```

b. 检查下载的二进制文件的 SHA-256 校验和。

```
openssl sha1 -sha256 kubectl
```

c. 确保输出中生成的校验和与下载的 kubectl.sha256 文件中的校验和相匹配。

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

Linux (amd64)

要在 Linux（**amd64**）上安装或更新 **kubectl**

1. 从 Amazon S3 为集群的 Kubernetes 版本下载 `kubectl` 二进制文件。
   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.29`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.28`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.27`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.26`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.25`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/amd64/kubectl
     ```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/linux/amd64/kubectl
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/linux/amd64/kubectl
```

2. （可选）使用二进制文件的 SHA-256 校验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 校验和。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 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes 1.30

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/linux/amd64/kubectl.sha256
```

b. 使用下面的命令之一检查下载的二进制文件的 SHA-256 校验和。

-

```
sha256sum -c kubectl.sha256
```

使用此命令时，请确保看到以下输出：

```
kubectl: OK
```

- ```
  openssl sha1 -sha256 kubectl
  ```

  使用此命令时，请确保输出中生成的校验和与下载的 kubectl.sha256 文件中的校验和相匹配。

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。

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

Linux（arm64）

要在 Linux（**arm64**）上安装或更新 **kubectl**

1. 从 Amazon S3 为集群的 Kubernetes 版本下载 kubectl 二进制文件。
   - Kubernetes 1.30
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.25`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.24`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.22`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/linux/arm64/kubectl
```

- Kubernetes `1.21`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/linux/arm64/kubectl
```

2. （可选）使用二进制文件的 SHA-256 校验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 校验和。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 `1.30`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/arm64/kubectl.sha256
   ```

   - Kubernetes `1.30`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/linux/arm64/kubectl.sha256
   ```

   - Kubernetes `1.29`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/linux/arm64/kubectl.sha256
   ```

   - Kubernetes `1.28`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/linux/arm64/kubectl.sha256
   ```

   - Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.22

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.21

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2024-04-19/bin/linux/arm64/kubectl.sha256
```

b. 使用下面的命令之一检查下载的二进制文件的 SHA-256 校验和。

- ```
  sha256sum -c kubectl.sha256
  ```

  使用此命令时，请确保看到以下输出：

  ```
  kubectl: OK
  ```
  ```

- ```
  openssl sha1 -sha256 kubectl
  ```

  使用此命令时，请确保输出中生成的校验和与下载的 `kubectl.sha256` 文件中的校验和相匹配。

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。

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

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 `1.30`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.0/2024-05-12/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.29`
     ```

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.3/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.28

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.8/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.12/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.26

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.25

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.24

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.23

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.22

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.17/2024-04-19/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.21

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.21.14/2024-04-19/bin/windows/amd64/kubectl.exe
```

3. （可选）使用二进制文件的 SHA-256 校验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 校验和。

   - Kubernetes 1.30

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.30.0/2024-05-12/bin/windows/amd64/kubectl.exe.sha256
     ```

   - Kubernetes 1.29

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.29.3/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
     ```

   - Kubernetes 1.28

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.28.8/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
     ```

   - Kubernetes 1.27

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.27.12/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
     ```

   - Kubernetes 1.26

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.26.15/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
     ```

   - Kubernetes 1.25

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.25.16/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
     ```

   - Kubernetes 1.24

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
     eks/1.24.17/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
     ```

- Kubernetes 1.23

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.23.17/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.22

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.22.17/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes 1.21

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.21.14/2024-04-19/bin/windows/amd64/kubectl.exe.sha256
```

b. 检查下载的二进制文件的 SHA-256 校验和。

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

c. 确保输出中生成的校验和与下载的 kubectl.sha256 文件中的校验和相匹配。PowerShell 的输出应为等效的大写字符串。

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。

3. 安装 kubectl 后，您可以验证其版本。

```
kubectl version --client
```

首次安装 kubectl 时，它尚未配置为与任何服务器通信。我们将根据需要在其他过程中介绍此配置。如果您需要更新配置以与特定集群通信，可以运行以下命令。将 *region-code* 替换为集群所在的 AWS 区域。将 *my-cluster* 替换为您的集群名称。

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

开始使用 Amazon EKS

在阅读入门指南之前，请确保您已设置好使用 Amazon EKS。有关更多信息，请参阅 设置使用 Amazon EKS。

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` – 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 – AWS Management Console 和 AWS CLI – 此入门指南可帮助您使用 AWS Management Console 和 AWS 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。

我们还提供以下参考资料：

- 有关精选的动手教程集合，请参阅 AWS 社区上的导航 Amazon EKS。
- 有关代码示例，请参阅使用 AWS SDK 的 Amazon EKS 代码示例。

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 AWS 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 AWS 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 – AWS Management Console 和 AWS CLI。

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。有关更多信息，请参阅 安装或更新 kubectl。
- **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 eksctl 文档中的 Installation。
- 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、服务相关角色、AWS CloudFormation、VPC 和相关资源的权限。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。要查看当前用户，请运行以下命令：

```
aws sts get-caller-identity
```

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 和 Amazon EKS 节点。一些设置仅在创建集群和节点时可以启用。

您可以使用下列节点类型之一创建集群。要了解有关各个类型的更多信息，请参阅 Amazon EKS 节点。部署集群后，您可以添加其他节点类型。

- Fargate - Linux - 如果要在 AWS Fargate 上运行 Linux 应用程序，请选择此类型的节点。Fargate 是一种无服务器计算引擎，允许部署 Kubernetes Pods，而不管理 Amazon EC2 实例。
- 托管式节点 – Linux – 如果要在 Amazon EC2 实例上运行Amazon Linux 应用程序，请选择此类型的节点。虽然本指南中未作介绍，但您还可以向集群添加 Windows 自行管理节点和 Bottlerocket 节点。

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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。将 *region-code* 替换为 Amazon EKS 支持的任一 AWS 区域。有关 AWS 区域列表，请参阅 AWS 一般参考中的 Amazon EKS 端点和配额。

Fargate – Linux

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

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 AWS CloudFormation 控制台 https://console.aws.amazon.com/cloudformation 中查看名为 eksctl-*my-cluster*-cluster 的 AWS 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
   ```

   示例输出如下。

   Fargate – Linux

   ```
   NAME                                                STATUS   ROLES    AGE
      VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-
   VERSION                   CONTAINER-RUNTIME
   fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
     8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
     1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
   fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
     7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
     1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
   ```

Managed nodes – Linux

```
NAME                                          STATUS   ROLES    AGE    VERSION
          INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
       CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
   v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
   v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

如需详细了解输出中显示的内容，请参阅 查看 Kubernetes 资源。

2. 查看在集群上运行的工作负载。

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

示例输出如下。

Fargate – Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
  192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
   <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
  192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
   <none>
```

Managed nodes – Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
          NODE                                          NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345             1/1     Running   0          7m43s
   192.0.2.1    ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

```
kube-system   aws-node-67890              1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-abcde    1/1     Running   0          14m
   192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345    1/1     Running   0          14m
   192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-12345            1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-67890            1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.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。

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。

```
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。

- 将示例应用程序部署到您的集群。
- 创建集群的 IAM 主体是唯一可以使用 kubectl 或 AWS Management Console 调用 Kubernetes API 服务器的主体。如果您希望其他 IAM 主体拥有访问您的集群的权限，您需要添加它们。有关更多信息，请参阅授予对 Kubernetes API 的访问权限 和所需的权限。
- 我们建议您先熟悉集群和节点的所有设置，然后再部署集群用于生产环境。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，请配置 Amazon VPC 容器网络接口插件以将 IAM 角色用于服务账户。

开始使用 Amazon EKS – AWS Management Console 和 AWS CLI

本指南帮助您创建通过 AWS Management Console 和 AWS 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`。

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。

- AWS CLI – 与AWS服务（包括 Amazon EKS）一起使用的命令行工具。有关更多信息，请参阅 AWS Command Line Interface 用户指南中的安装、更新和卸载 AWS CLI。在安装 AWS CLI 后，建议您还要对其进行配置。有关更多信息，请参阅 AWS 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。有关更多信息，请参阅 安装或更新 `kubectl`。
- 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、服务相关角色、AWS CloudFormation、VPC 和相关资源的权限。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。要查看当前用户，请运行以下命令：

  ```
  aws sts get-caller-identity
  ```

- 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。此外，您的 Shell 的引用和转义规则可能有所不同。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的在 AWS CLI 中将引号和字符串结合使用。

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。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 AWS 区域。有关 AWS 区域 列表，请参阅 AWS 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.us-west-2.amazonaws.com/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 AWS 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 AWS 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 AWS 区域 是您要在其中创建集群的AWS 区域。如果不是，请选择 AWS 区域名称旁边的下拉菜单，然后选择要使用的 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。

   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. 在配置可观测性页面上，选择下一步。

8. 在选择附加组件页中，选择下一个。

   有关附加组件的更多信息，请参阅 Amazon EKS 附加组件。

9. 在配置选定的附加组件设置页面上，选择下一个。
10. 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。

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 AWS 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，请参阅故障排除主题中的 未经授权或访问被拒绝 (kubectl)。

    示例输出如下。

```
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。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之一创建集群。要了解有关各个类型的更多信息，请参阅 Amazon EKS 节点。部署集群后，您可以添加其他节点类型。

- Fargate - Linux - 如果要在 AWS Fargate 上运行 Linux 应用程序，请选择此类型的节点。Fargate 是一种无服务器计算引擎，允许部署 Kubernetes Pods，而不管理 Amazon EC2 实例。
- 托管节点 – Linux – 如果要运行，请选择此类型的节点 Amazon Linux 应用程序。虽然本指南中未作介绍，但您还可以向集群添加 Windows 自行管理节点和 Bottlerocket 节点。

Fargate – Linux

创建 Fargate 配置文件。当部署 Kubernetes Pods 时使用的条件符合配置文件中定义的条件时，这些 Pods 将会部署到 Fargate。

创建 Fargate 配置文件的步骤

1. 创建 IAM 角色并向其附加所需的 Amazon EKS IAM 托管策略。当您的集群在 Fargate 基础设施上创建 Pods 时，在 Fargate 基础设施上运行的组件必须代表您调用 AWS 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 AWS 服务的操作。Amazon EKS Pod 执行角色提供执行此操作的 IAM 权限。

   a. 将以下内容复制到名为 *pod-execution-role-trust-policy*.json 的文件中。请将 *region-code* 替换为集群所在的 AWS 区域。如果您希望在账户中在所有 AWS 区域 使用相同角色，请将 *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. 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"
```

c. 将所需的 Amazon EKS 托管 IAM 策略附加到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/
AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

2. 访问 https://console.aws.amazon.com/eks/home#/clusters 打开 Amazon EKS 控制台。
3. 在 Clusters（集群）页面上，选择 *my-cluster* 集群。
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. 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，请执行以下操作：

   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），选择您在上一步中创建的 AmazonEKSFargatePodExecutionRole。

   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. 选择下一步。

6. 在 Configure Pod selection（配置选择）页面上，请执行以下操作：

   a. 对于命名空间，请输入 **`default`**。

   b. 选择下一步。

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）。

8. 几分钟后，Fargate Profile configuration（Fargate 配置文件配置）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。

9. 如果您计划将所有 Pods 部署到 Fargate（不部署到 Amazon EC2 节点），请执行以下操作以创建另一个 Fargate 配置文件，并在 Fargate 上运行默认名称解析器 (CoreDNS)。

   > **Note**
   >
   > 否则，您目前会没有任何节点。

   a. 请在 Fargate Profile（Fargate 配置文件）页面上，选择 *`my-profile`*。

   b. 在 Fargate profiles（Fargate 配置文件）下，选择 Add Fargate profile（添加 Fargate 配置文件）。

   c. 对于名称，请输入 ***`CoreDNS`***。

   d. 对于 Pod execution role（容器组（pod）执行角色），选择您在上一步中创建的 AmazonEKSFargatePodExecutionRole。

   e. 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。

f. 选择下一步。

g. 对于命名空间，请输入 **kube-system**。

h. 请选择 Match labels（匹配标签），然后选择 Add label（添加标签）。

i. 输入 **k8s-app** 作为 Key（密钥），输入 **kube-dns** 作为值。必须这样设置，默认的名称解析器（CoreDNS）才能部署到 Fargate。

j. 选择下一步。

k. 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）。

l. 运行以下命令以从 CoreDNS Pods 中删除默认的 `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"}]'
```

> **Note**
>
> 系统根据您添加的 Fargate 配置文件标签创建和部署两个节点。由于它们不适用于 Fargate 节点，因此您不会看到 Node groups（节点组）中列出任何内容，但将看到 Overview（概览）选项卡中列出新节点。

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 AWS 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 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***。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。

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。

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或 Fargate 配置文件。

   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. 如果您创建了节点组，请 *my-nodegroup* 节点组，然后选择 Delete（删除）。输入 ***my-nodegroup***，然后选择删除。

e. 请选择您创建的每个 Fargate 配置文件，然后选择 Delete（删除）。请输入配置文件的名称，然后选择 Delete（删除）。

> **Note**
>
> 删除第二个 Fargate 配置文件时，您可能需要等待第一个配置文件完成删除。

f. 在节点组或 Fargate 配置文件删除后再继续操作。

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 AWS CloudFormation 堆栈。

a. 打开 AWS 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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

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 AmazonEKSFargatePodExecutionRole 或 *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 主体是唯一可以使用 `kubectl` 或 AWS Management Console 调用 Kubernetes API 服务器的主体。如果您希望其他 IAM 主体拥有访问您的集群的权限，您需要添加它们。有关更多信息，请参阅授予对 Kubernetes API 的访问权限 和所需的权限。
- 将示例应用程序部署到您的集群。
- 在部署集群用于生产用途前，我们建议您熟悉集群和节点的所有设置。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，请配置 Amazon VPC 容器网络接口插件以将 IAM 角色用于服务账户。

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 AWS 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 AWS 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 8GiB。通过运行以下命令，可以监控当前数据库大小的指标。如果您的集群有低于 `1.28` 的 Kubernetes 版本，请将 *`apiserver_storage_size_bytes`* 替换为以下版本：

- Kubernetes 版本 `1.27` 和 `1.26` – **`apiserver_storage_db_total_size_in_bytes`**
- Kubernetes 版本 `1.25` 及以下 – **`etcd_db_total_size_in_bytes`**

```
kubectl get --raw=/metrics | grep "apiserver_storage_size_bytes"
```

Amazon EKS 节点在AWS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 中的工作方式。

- 有关互联的集群，请参阅 Amazon EKS Connector。

主题

- 创建 Amazon EKS 集群
- 集群见解
- 更新 Amazon EKS 集群 Kubernetes 版本
- 删除 Amazon EKS 集群
- Amazon EKS 集群端点访问控制
- 在现有集群中启用密钥加密
- 为 Amazon EKS 集群启用 Windows 支持
- 私有集群要求
- Amazon EKS Kubernetes 版本
- Amazon EKS 平台版本
- Autoscaling

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。如果您需要在 AWS Outpost 上创建集群，请参阅 AWS Outposts 上的 Amazon EKS 的本地集群。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 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 AWS CloudFormation 模板创建它们。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --`**

`version | cut -d / -f2 | cut -d ' ' -f1`。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

- 具有 `create` 和 `describe` Amazon EKS 集群权限的 IAM 主体。有关更多信息，请参阅在 Outpost 上创建本地 Kubernetes 集群 和列出或描述所有集群。

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 主体分配 `iam:CreateRole` 操作（权限）。

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

3. 您可以分配 Amazon EKS 托管策略或创建自己的自定义策略。有关必须在自定义策略中使用的最低权限，请参阅 Amazon EKS 集群 IAM 角色。

   将名为 AmazonEKSClusterPolicy 的 Amazon EKS 托管 IAM 策略附加到角色。要将 IAM policy 附加到某个 IAM 主体，必须为附加该策略的主体分配以下 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、AWS Management Console 或 AWS CLI 创建集群。

   eksctl

   先决条件

   您的设备或 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。

   创建集群的步骤

   在您的默认 AWS 区域 中，使用 Amazon EKS 默认 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 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - 将 *1.29* 替换为 Amazon EKS 支持的版本。

     > **Note**
     >
     > 要在此时部署 1.30 集群，您需要使用 AWS Management Console 或 AWS CLI。

   - 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。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。

```
eksctl create cluster --name my-cluster --region region-code --version 1.29 --
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”。您可以修改 Amazon EKS 创建的集群安全组中的规则。
- 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。

- 如果您创建集群并希望集群分配 IPv6 地址而不是 IPv4 地址到 Pods 和服务，则请指定 ipFamily 选项。

  默认情况下，Kubernetes 将 IPv4 地址分配给 Pods 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”、the section called “子网要求和注意事项”、the section called “安全组要求” 和 the section called “IPv6” 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。

AWS 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。名称只能包含字母数字字符（区分大小写）、连字符和下划线。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 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。建议选择最新版本，除非您需要早期版本。
   - 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 AWS 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，请确保您熟悉在现有集群中启用密钥加密 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。

  完成此页面后，请选择下一步。

4. 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 Amazon EKS VPC 和子网要求和注意事项 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 为 Amazon EKS 集群创建 VPC。
   - 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。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。

     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”。您可以修改 Amazon EKS 创建的集群安全组中的规则。
   - 选择集群 IP 地址系列 - 您可以选择 IPv4 和 IPv6。

     默认情况下，Kubernetes 将 `IPv4` 地址分配给 Pods 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”、the section called “子网要求和注意事项”、the section called “安全组要求” 和 the section called “IPv6” 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 Amazon EKS 集群端点访问控制。

完成此页面后，请选择下一步。

5. （可选）在配置可观测性页面上，选择要开启的指标和控制面板日志记录选项。默认情况下，每种日志类型都处于关闭状态。
   - 有关 Prometheus 指标选项的更多信息，请参阅 创建集群时开启 Prometheus 指标。
   - 有关控制面板日志记录选项的更多信息，请参阅 Amazon EKS 控制面板日志记录。

   完成此页面后，请选择下一步。
6. 在 Select add-ons（选择附加组件）页面上，选择要添加到集群的附加组件。您可以根据需要选择任意数量的 Amazon EKS 附加组件和 AWS Marketplace 附加组件。如果未列出要安装的 AWS Marketplace 附加组件，则可以通过在搜索框中输入文本来搜索可用的 AWS Marketplace 附加组件。您也可以按 category（类别）、vendor（供应商）或 pricing model（定价模式）进行搜索，然后从搜索结果中选择附加组件。完成此页面后，请选择下一步。
7. 在配置选定插件设置页面上，选择要安装的版本。创建集群后，您可以随时更新到更高版本。创建集群后，您可以更新每个附加组件的配置。有关配置附加组件的更多信息，请参阅更新附加组件。完成此页面后，请选择下一步。
8. 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。

集群预配置需要几分钟时间。

AWS 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 AWS 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）、连字符和下划线。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - 将 *1.30* 替换为 Amazon EKS 支持的版本。
   - 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。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。
   - 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”。您可以修改 Amazon EKS 创建的集群安全组中的规则。

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-
version 1.30 \
   --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。

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。
- 如果您创建集群并希望集群分配 Pods 地址而不是 IPv4 地址到 IPv6 和服务，请将 **--kubernetes-network-config ipFamily=ipv6** 添加到以下命令中。

  默认情况下，Kubernetes 将 IPv4 地址分配给 Pods 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”、the section called “子网

要求和注意事项”、the section called “安全组要求” 和 the section called “IPv6” 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。

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 为 Amazon EKS 集群创建或更新 kubeconfig 文件。

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

示例输出如下。

```
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
```

示例输出如下。

```
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 AWS Identity and Access Management（IAM）权限，请为集群创建 IAM OpenID Connect（OIDC）提供商。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。要了解有关将特定 IAM 权限分配给工作负载的详情，请参阅 服务账户的 IAM 角色。

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

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 配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户（IRSA） 或 Amazon EKS 节点 IAM 角色。

7. 如果您使用 AWS Management Console 部署了集群，则可以跳过此步骤。AWS 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。

   如果您使用 eksctl 或 AWS 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。

8. （可选）如果您尚未执行此操作，可以为集群启用 Prometheus 指标。有关更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的创建抓取程序。

9. 如果启用了 Prometheus 指标，则必须设置 aws-auth ConfigMap 以授予抓取程序集群内权限。有关更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的配置您的 Amazon EKS 集群。

10. 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 Amazon EBS CSI 驱动程序 安装到您的集群。

向您建议的后续步骤：

- 创建集群的 IAM 主体是唯一可以访问集群的主体。向其他 IAM 主体授予权限，以便它们可以访问您的集群。
- 如果创建集群的 IAM 主体仅具有先决条件中引用的最低 IAM 权限，则您可能想要为该主体添加额外的 Amazon EKS 权限。有关向 IAM 主体授予 Amazon EKS 权限的更多信息，请参阅适用于 Amazon EKS 的身份和访问管理。

- 如果您希望创建集群的 IAM 主体或任何其他主体在 Amazon EKS 控制台中查看 Kubernetes 资源，请向实体授予 所需的权限。
- 如果您希望节点和 IAM 主体从 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 Amazon EKS 集群端点访问控制。
- 为集群启用密钥加密。
- 为集群配置日志记录。
- 将节点添加到集群。

集群见解

Amazon EKS 集群见解提供建议，帮助您遵循 Amazon EKS 和 Kubernetes 最佳实践。每个 Amazon EKS 集群都会根据 Amazon EKS 精心策划的见解列表进行自动的定期检查。这些见解检查完全由 Amazon EKS 管理，并就如何解决任何调查发现提供建议。

集群见解的建议用法：

- 在更新集群 Kubernetes 版本之前，请在 EKS 控制台中查看集群见解。
- 如果您的集群已发现问题，请查看它们并进行适当的修复。这些问题包括指向 Amazon EKS 和 Kubernetes 的链接。
- 修复问题后，等待集群见解刷新。如果所有问题都已解决，则请更新您的集群。

目前，Amazon EKS 仅返回与 Kubernetes 版本升级准备情况相关的见解。

升级见解可以识别可能影响 Kubernetes 集群升级的可能问题。这样可以最大限度地减少管理员准备升级所花费的工作量，并提高新 Kubernetes 版本上应用程序的可靠性。Amazon EKS 会根据可能影响 Kubernetes 版本升级的问题列表自动扫描集群。Amazon EKS 经常根据对每个 Kubernetes 版本中所做更改的审查来更新见解检查列表。

Amazon EKS 升级见解加快了新版本的测试和验证过程。还允许集群管理员和应用程序开发人员通过突出问题和提供补救建议来利用最新 Kubernetes 功能。要查看已执行的见解检查列表以及 Amazon EKS 发现的任何相关问题，您可以调用 Amazon EKS `ListInsights` API 操作或在 Amazon EKS 控制台中查看。

集群见解将定期更新。您无法手动刷新集群见解。如果您修复集群问题，则将需要一些时间才能更新集群见解。要确定修复是否成功，请将更改部署的时间与集群洞察的“上次刷新时间”进行比较。

查看集群见解（控制台）

要查看 Amazon EKS 集群的见解：

a. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。

b. 从集群列表中，选择您希望查看见解的 Amazon EKS 集群的名称。

c. 选择升级见解选项卡。

d. 在升级见解页面上，您将看到以下字段：

- 名称 – Amazon EKS 对集群执行的检查。
- 见解状态 – 状态为“错误”的见解通常表示受影响的 Kubernetes 版本是当前集群版本的 N+1，而状态为“警告”表示该见解适用于未来 Kubernetes 版本 N+2 或更高版本。状态为“通过”的见解表示 Amazon EKS 在您的集群中未发现与该见解检查相关的任何问题。状态为“未知”的见解表示 Amazon EKS 无法确定您的集群是否受到此见解检查的影响。
- 版本 – 见解检查可能存在的问题的 Kubernetes 版本。
- 上次刷新时间（UTC-5:00）– 此集群上次刷新见解状态的时间。
- 上次转换时间（UTC-5:00）– 上次更改此见解状态的时间。
- 描述 – 来自见解检查的信息，包括警报和建议的补救措施。

查看集群见解（AWS CLI）

要查看 Amazon EKS 集群的见解：

a. 确定您要检查哪个集群以获取见解。以下命令列出指定集群的见解。根据需要对该命令进行以下修改，然后运行修改后的命令：

- 将 *region-code* 替换为您的 AWS 区域 的代码。
- 将 *my-cluster* 替换为您的集群名称。

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

示例输出如下。

```
{
    "insights": [
        {
            "category": "UPGRADE_READINESS",
            "name": "Deprecated APIs removed in Kubernetes v1.29",
```

```
            "insightStatus": {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30 days."
            },
            "kubernetesVersion": "1.29",
            "lastTransitionTime": 1698774710.0,
            "lastRefreshTime": 1700157422.0,
            "id": "123e4567-e89b-42d3-a456-579642341238",
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes v1.29. Upgrading your cluster before migrating to the
 updated APIs supported by v1.29 could cause application impact."
        }
    ]
}
```

b. 有关见解的描述性信息，请运行以下命令。根据需要对该命令进行以下修改，然后运行修改后的命令：

- 将 *region-code* 替换为您的 AWS 区域 的代码。
- 将 *123e4567-e89b-42d3-a456-579642341238* 替换为从列出集群见解中检索到的见解 ID。
- 将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-insight --region region-code --id 123e4567-e89b-42d3-
a456-579642341238 --cluster-name my-cluster
```

示例输出如下。

```
{
    "insight": {
        "category": "UPGRADE_READINESS",
        "additionalInfo": {
            "EKS update cluster documentation": "https://docs.aws.amazon.com/eks/
latest/userguide/update-cluster.html",
            "Kubernetes v1.29 deprecation guide": "https://kubernetes.io/docs/
reference/using-api/deprecation-guide/#v1-29"
        },
        "name": "Deprecated APIs removed in Kubernetes v1.29",
        "insightStatus": {
            "status": "PASSING",
            "reason": "No deprecated API usage detected within the last 30 days."
        },
        "kubernetesVersion": "1.29",
```

```
        "recommendation": "Update manifests and API clients to use newer Kubernetes
 APIs if applicable before upgrading to Kubernetes v1.29.",
        "lastTransitionTime": 1698774710.0,
        "lastRefreshTime": 1700157422.0,
        "categorySpecificSummary": {
            "deprecationDetails": [
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
flowschemas",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
flowschemas",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                },
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
prioritylevelconfigurations",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
prioritylevelconfigurations",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                }
            ]
        },
        "id": "f6a11fe4-77f7-48c6-8326-9a13f022ecb3",
        "resources": [],
        "description": "Checks for usage of deprecated APIs that are scheduled for
 removal in Kubernetes v1.29. Upgrading your cluster before migrating to the updated
 APIs supported by v1.29 could cause application impact."
    }
}
```


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**
>
> 升级集群后，就无法降级到以前的版本。我们建议您在更新到新的 Kubernetes 版本之前，先查看 Amazon EKS Kubernetes 版本 中的信息，同时查看此主题中的更新步骤。

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。但是，一旦开始集群升级，就不能暂停或停止。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 个可用的 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。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**
>
> 为确保集群的 API 服务器端点始终可访问，Amazon EKS 提供了高度可用的 Kubernetes 控制面板，并且会在更新操作期间执行 API 服务器实例的滚动更新。由于支持 Kubernetes API 服务器端点的 API 服务器实例 IP 地址会持续变化，您必须确保您的 API 服务器客户端能够有效地管理重新连接。最新版本的 `kubectl` 和 Kubernetes 客户端库 已经获得正式支持，能够透明地执行此重新连接过程。

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。
   - 获取集群控制面板的 Kubernetes 版本。

```
kubectl version
```

- 获取您的节点的 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 1.29 版本，且其中一个节点正在运行 1.28 版本，则您必须将节点更新为 1.29 版本，然后将控制面板更新为 1.30。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 和自行管理节点的更新。如果 Fargate 节点的次要版本低于控制面板版本，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. 如果您最初部署集群所用的 Kubernetes 版本是 Kubernetes 1.25 或更高版本，请跳过此步骤。

   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Amazon EKS 默认 Pod 安全策略，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、AWS Management Console 或 AWS CLI 更新您的集群。

> **Important**
>
> - 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Kubernetes 1.23 和Amazon EBS CSI 驱动程序。
> - Kubernetes 1.24 及更高版本将 containerd 用作默认容器运行时系统。如果您要切换到 containerd 运行时系统并且已经为 Container Insights 进行了 Fluentd 配置，则必须先将 Fluentd 迁移到 Fluent Bit，然后才能更新集群。Fluentd 解析器配置为仅解析 JSON 格式的日志消息。与 dockerd 不同的是，containerd 容器运行时系统的日志消息不是 JSON 格式的。如果您不迁移到 Fluent Bit，一些配置的 Fluentd's 解析器将在 Fluentd 容器内生成大量错误。有关迁移的更多信息，请参阅将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs。
> - 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版本 1.28，而且您想将其更新到版本 1.30。您必须先将版本 1.28 集群更新为版本 1.29，然后将版本 1.29 集群更新为版本 1.30。
> - 查看节点上 Kubernetes kube-apiserver 和 kubelet 之间的版本偏差。
>   - 从 Kubernetes 版本 1.28 开始，kubelet 最多可能有三个早于 kube-apiserver 的次要版本。请参阅 Kubernetes 上游版本偏差策略。
>   - 如果您的托管节点和 Fargate 节点上的 kubelet 为 Kubernetes 版本 1.25 或更新版本，则无需更新 kubelet 版本即可将集群最多提前更新三个版本。例如，如果 kubelet 为版本 1.25，则可以在 kubelet 保持版本 1.25 的情况下，将您的 Amazon EKS 集群版本从 1.25 更新到 1.27、1.26 和 1.28。
>   - 如果您的托管节点和 Fargate 节点上的 kubelet 为 Kubernetes 版本 1.24 或更早版本，则最多只能有两个早于 kube-apiserver 的次要版本。换句话说，如果 kubelet 为版本 1.24 或更早的版本，则最多只能将集群提前更新两个版本。例如，

如果 kubelet 为版本 1.21，则可以将您的 Amazon EKS 集群版本从 1.21 更新到 1.22 和 1.23，但是当 kubelet 保持 1.21 时，您无法将集群更新到 1.24。

- 作为开始更新之前的最佳实践，请确保节点上的 kubelet 与控制面板具有相同的 Kubernetes 版本。
- 如果集群配置有早于 1.8.0 的 Amazon VPC CNI plugin for Kubernetes 版本，则建议您将插件更新为最新版本，然后再更新集群。要更新插件，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。
- 如果您要将集群更新到版本 1.25 或更高版本并将 AWS Load Balancer Controller 部署在集群中，则在将集群版本更新为 1.25 之前，将控制器更新到版本 2.4.7 或更高版本。有关更多信息，请参阅 Kubernetes 1.25 版本注释。

eksctl

此过程需要 eksctl 版本 0.183.0 或更高版本。可以使用以下命令来查看您的版本：

```
eksctl version
```

有关安装和更新 eksctl 的说明，请参阅 eksctl 文档中的 Installation。

更新 Amazon EKS 控制面板的 Kubernetes 版本。将 *my-cluster* 替换为您的集群名称。将 *1.30* 替换为您想要将集群更新到的 Amazon EKS 支持的版本号。有关支持的版本号列表，请参阅 Amazon EKS Kubernetes 版本。

```
eksctl upgrade cluster --name my-cluster --version 1.30 --approve
```

更新过程可能需要几分钟才能完成。

AWS 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. 对于集群名称，输入您的集群的名称并选择确认。

更新过程可能需要几分钟才能完成。

AWS CLI

a. 使用以下 AWS CLI 命令更新 Amazon EKS 集群。将 *example values* 替换为您自己的值。将 *1.30* 替换为您想要将集群更新到的 Amazon EKS 支持的版本号。有关支持的版本号列表，请参阅 Amazon EKS Kubernetes 版本。

```
aws eks update-cluster-version --region region-code --name my-cluster --
kubernetes-version 1.30
```

示例输出如下。

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.30"
            },
            {
                "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
```

示例输出如下。

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.30"
            },
            {
                "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 和更新托管节点组。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 `1.30`，则查找以 `1.30` 开头的 Cluster Autoscaler 最新版本。记录该版本的语义版本号（例如 `1.30.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 *`1.30.n`* 替换为您自己的值。

   ```
   kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:v1.30.n
   ```

7. （仅限具有 GPU 节点的集群）如果您的集群具有支持 GPU 的节点组（例如，`p3.2xlarge`），则您必须在集群上更新 适用于 Kubernetes 的 NVIDIA 设备插件 DaemonSet。将 *`vX.X.X`* 替换为您需要的 NVIDIA/k8s-device-plugin 版本，然后运行以下命令。

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

8. 更新 Amazon VPC CNI plugin for Kubernetes、CoreDNS 和 kube-proxy 附加组件。建议将附件组件更新为服务账户令牌中列出的最低版本。

   - 如果您正在使用 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您更新的集群的名称。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，选择立即更新，选择一个可用版本，然后选择更新。
   - 此外，您也可以使用 AWS CLI 或 eksctl 更新附加组件。有关更多信息，请参阅 更新附加组件。

9. 如有必要，请更新您的 kubectl 版本。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，1.29 kubectl 客户端使用 Kubernetes 1.28、1.29 和 1.30 集群。您可以使用以下命令检查当前已安装的版本。

```
kubectl version --client
```

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

> **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。
> - 适用于 Prometheus 的 Amazon 托管服务资源不在集群生命周期内，需要独立于集群进行维护。删除集群时，请务必同时删除所有适用的抓取器以停止适用的费用。有关更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的查找和删除抓取程序。

您可以使用 eksctl、AWS Management Console 或 AWS CLI 删除集群。

eksctl

使用 **eksctl** 删除 Amazon EKS 集群和节点

此过程需要 eksctl 版本 0.183.0 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 的说明，请参阅 eksctl 文档中的 Installation。

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
   ```

AWS Management Console

使用 AWS 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。

   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 AWS CloudFormation 堆栈。

   a. 打开 AWS 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 AWS CloudFormation 堆栈。

   a. 打开 AWS 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）。

AWS CLI

使用 AWS 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。

   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 AWS CloudFormation 堆栈。

   a. 使用以下命令列出您的可用 AWS 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 AWS CloudFormation 堆栈。

a. 使用以下命令列出您的可用 AWS 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 AWS 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 AWS API 通信的传统 AWS 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
已启用	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

终端节点公有访问	终端节点私有访问	行为
		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。
已启用	已启用	• 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。

终端节点公有访问	终端节点私有访问	行为
已禁用	已启用	• 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。 • 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。 • 更新您的集群。

您可以使用 AWS Management Console或 AWS CLI 修改集群 API 服务器终端节点访问。

AWS Management Console

使用 AWS 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。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。

AWS CLI

使用 AWS CLI 修改集群 API 服务器端点访问

使用 AWS CLI 版本 `1.27.160` 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 AWS CLI，请参阅安装 AWS CLI。

1. 使用下面的 AWS 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。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
```

示例输出如下。

```
{
    "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
```

示例输出如下。

```
{
    "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 AWSTransit Gateway 或其他连接选项将您的网络连接到 VPC，然后使用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 AWS 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。

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 AWS 凭证，或在移除端点公有访问之前添加您的堡垒机将用于 RBAC 配置的 IAM 主体。有关更多信息，请参阅the section called “授予对 Kubernetes API 的访问权限” 和未经授权或访问被拒绝 (`kubectl`)。

AWS Cloud9 IDE

AWS Cloud9 是一种基于云的集成式开发环境（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 AWS 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 AWS 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。

在为 AWS Cloud9 IDE 配置 `kubectl` 时，请确保使用已映射到集群的 RBAC 配置的 AWS 凭证，或在移除端点公有访问之前添加您的 IDE 将用于 RBAC 配置的 IAM 主体。有关更多信息，请参阅授予对 Kubernetes API 的访问权限 和未经授权或访问被拒绝 (`kubectl`)。

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 AWS 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 AWS 区域 中创建
- 如果 KMS 密钥是在其他账户中创建的，则 IAM 主体必须拥有对 KMS 密钥的访问权限。

有关更多信息，请参阅《AWS Key Management Service 开发人员指南》https://docs.aws.amazon.com/kms/latest/developerguide/中的允许其他账户中的 IAM 主体使用 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 `kms-cluster.yaml` 文件向集群添加加密。

  ```
  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
```

AWS 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。

AWS CLI

1. 使用以下 AWS 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"}}]'
```

示例输出如下。

```
{
  "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
```

示例输出如下。

```
{
  "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 AWS 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 AWS 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。

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

- 您可以使用 `HostProcess` 容器组（pod）在 Windows 节点上使用主机网络。有关更多信息，请参阅 Kubernetes 文档中的创建 Windows `HostProcessPod`。
- 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 200MB 限制。
- 您不能将 Pods 的安全组 用于在 Windows 节点上运行的 Pods。
- 您不能将自定义联网用于 Windows 节点。
- 您不可以将 `IPv6` 与 Windows 节点一起使用。
- Windows 节点支持每个节点一个弹性网络接口。默认情况下，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《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。
- 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。
- Fargate 上的 Amazon EKS Pods 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。
- 为 Windows 托管节点组指定自定义 AMI ID 时，请将 `eks:kube-proxy-windows` 添加到您的 AWS IAM 身份验证器配置映射中。有关更多信息，请参阅 指定 AMI ID 时的限制和条件。

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。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 支持并启用 Windows 支持。

Kubernetes 版本	平台版本
1.30	eks.2
1.29	eks.1
1.28	eks.1

Kubernetes 版本	平台版本
1.27	eks.1
1.26	eks.1
1.25	eks.1
1.24	eks.2

- 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。

启用 Windows 支持

如果您的集群不是先决条件部分中列出的 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。

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。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。将 *eksClusterRole* 替换为您的集群角色名称。

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

   示例输出如下。

   ```
   {
       "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。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 应用于集群。

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

5. 验证 aws-auth ConfigMap 是否包含 Windows 节点的实例角色映射，以包含 eks:kube-proxy-windows RBAC 权限组。您可以通过运行以下命令进行验证。

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

示例输出如下。

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

```
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws:iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

您应该会看到组下面列出的 eks:kube-proxy-windows。如果未指定组，则需要更新 ConfigMap 或进行创建，以包含所需的组。有关 aws-auth ConfigMap 的更多信息，请参阅 将 aws-authConfigMap 应用到集群。

从数据面板中删除旧版 Windows 支持

如果您在早于先决条件列出的 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 AWS 区域。

```
kubectl delete -f https://s3.us-west-2.amazonaws.com/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 AWS 区域。

```
kubectl delete -f https://s3.us-west-2.amazonaws.com/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。

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，请跳过此步骤。

   从您的集群角色中删除 AmazonVPCResourceController 这项托管的 IAM 策略。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，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。

如果您的集群或平台版本早于先决条件部分中列出的版本，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 支持并为控制面板启用新版支持。

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 `0.183.0` 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 `eksctl` 文档中的 Installation。

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。

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。

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 AWS 区域。

1. 将 VPC 资源控制器部署到您的集群。

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/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 https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/Setup-VPCAdmissionWebhook.ps1;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
curl -O https://s3.us-west-2.amazonaws.com/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。

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。

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 AWS 区域。

1. 将 VPC 资源控制器部署到您的集群。

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/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 https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-bundle.sh
curl -O https://s3.us-west-2.amazonaws.com/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。

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。

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
```

示例输出如下。

```
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 https://s3.us-west-2.amazonaws.com/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 https://s3.us-west-2.amazonaws.com/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。

在 Windows 节点上支持更高的 Pod 密度

在 Amazon EKS 中，每个 Pod 都会从您的 VPC 分配一个 IPv4 地址。因此，即使有足够的资源可以在节点上运行更多 Pods，可以部署到该节点的 Pods 数量也受到可用 IP 地址的限制。由于 Windows 节点仅支持一个弹性网络接口，因此默认情况下，Windows 节点上可用 IP 地址的最大数量等于：

```
Number of private IPv4 addresses for each interface on the node - 1
```

一个 IP 地址用作网络接口的主要 IP 地址，因此无法将其分配给 Pods。

通过启用 IP 前缀委派，可以在 Windows 节点上启用更高的 Pod 密度。此功能使您可以为主网络接口分配 /28 IPv4 前缀，而不是分配辅助 IPv4 地址。分配 IP 前缀会将节点上的最大可用 IPv4 地址数增加到：

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

由于可用 IP 地址的数量要多得多，可用的 IP 地址不应限制您在节点上扩展 Pods 数量的能力。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量。

私有集群要求

本主题将介绍如何部署部署在 AWS Cloud 上但没有出站互联网访问权限的 Amazon EKS 集群。如果您在 AWS Outposts 上有一个本地集群，请参阅 在 Outpost 上启动自行管理的 Amazon Linux 节点 而不是本主题。

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。如果集群没有出站互联网访问权限，则其必须满足以下要求：

- 您的集群必须从 VPC 中的容器注册表中拉取映像。您可以在 VPC 中创建 Amazon Elastic Container Registry，并将容器映像复制到其中，以供节点拉取。有关更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库。
- 集群必须启用端点私有访问权限。需要启用该访问权限，节点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。
- 自行管理 Linux 和 Windows 节点必须包含以下引导参数才能进行启动。这些实际参数绕过 Amazon EKS 自检，且无需从 VPC 内访问 Amazon EKS API。

  1. 使用以下命令确定集群端点的值。将 *my-cluster* 替换为您集群的名称。

     ```
     aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
     ```

     示例输出如下。

     ```
     https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
     ```

  2. 使用以下命令确定集群证书颁发机构的值。将 *my-cluster* 替换为您集群的名称。

     ```
     aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --output text
     ```

     返回的输出是一个长字符串。

  3. 请将以下命令中的 *cluster-endpoint* 和 *certificate-authority* 替换为上一个命令的输出中返回的值。有关在启动自行管理的节点时指定引导参数的更多信息，请参阅 启动自行管理的 Amazon Linux 节点 和 启动自行管理的 Windows 节点。

  - 对于 Linux 节点：

    ```
    --apiserver-endpoint cluster-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：

    > **Note**
    >
    > 如果您使用自定义服务 CIDR，则需要使用 `-ServiceCIDR` 参数进行指定。否则，集群 Pods 中的 DNS 解析将失败。

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

有关更多参数，请参阅 引导脚本配置参数。

- 必须从 VPC 内创建集群的 aws-auth ConfigMap。有关在 aws-auth ConfigMap 中创建和添加条目的更多信息，请在终端中输入 **eksctl create iamidentitymapping --help**。如果您的服务器上不存在 ConfigMap，则 eksctl 将在您使用命令添加身份映射时创建。
- 配置了服务账户 IAM 角色的 Pods 通过 AWS Security Token Service（AWS STS）API 调用获取凭证。如果没有出站互联网访问权限，则您必须在 VPC 中创建并使用 AWS STS VPC 端点。预设情况下，大多数 AWS v1 开发工具包均使用全局 AWS STS 端点（sts.amazonaws.com），而此端点不使用 AWS STS VPC 端点。要使用 AWS STS VPC 端点，您可能需要将开发工具包配置为使用区域 AWS STS 端点（sts.*region-code*.amazonaws.com）。有关更多信息，请参阅 为服务账户配置 AWS Security Token Service 端点。

- 集群的 VPC 子网必须具有 Pods 需要访问的任何 AWS 服务 的 VPC 接口端点。有关更多信息，请参阅使用接口 VPC 端点访问 AWS 服务。下表列出了一些常用的服务和端点。有关端点的完整列表，请参阅《AWS PrivateLink 指南》中的 与 AWS PrivateLink 集成的 AWS 服务。

服务	终端节点
Amazon EC2	com.amazonaws.*region-code* .ec2
Amazon Elastic Container Registry（用于拉取容器映像）	com.amazonaws.*region-code* .ecr.api、com.amazonaws.*region-code* .ecr.dkr 和 com.amazonaws.*region-code* .s3
Application Load Balancer 和 Network Load Balancer	com.amazonaws.*region-code* .elasticloadbalancing
AWS X-Ray	com.amazonaws.*region-code* .xray
Amazon CloudWatch Logs	com.amazonaws.*region-code* .logs
AWS Security Token Service（将 IAM 角色用于服务账户时为必需）	com.amazonaws.*region-code* .sts

注意事项

- 任何自行管理的节点都必须部署到具有您需要的 VPC 接口端点的子网中。如果您创建托管节点组，则 VPC 接口端点安全组必须允许子网的 CIDR，或者您必须将已创建的节点安全组添加到 VPC 接口端点安全组。
- 如果您的 Pods 使用 Amazon EFS 卷，那么在部署 Amazon EFS CSI 驱动程序 之前，必须更改驱动程序的 kustomization.yaml 文件，以将容器映像设置为使用与 Amazon EKS 集群相同的 AWS 区域。
- 您可以使用 AWS Load Balancer Controller 将 AWS Application Load Balancers（ALB）和 Network Load Balancers 部署到私有集群。部署时，您应该使用命令行标记将 `enable-shield`、`enable-waf` 和 `enable-wafv2` 设置为 false。带有来自 Ingress 对象中的主机名的证书发现不受支持。这是因为控制器需要访问没有 VPC 接口端点的 AWS Certificate Manager。

  该控制器支持带有 IP 目标的网络负载均衡器，结合 Fargate 一起使用时需要网络负载均衡器。有关更多信息，请参阅Amazon EKS 上的应用程序负载均衡 和创建网络负载均衡器。
- 支持 Cluster Autoscaler。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 AWS STS VPC 端点和自动扩展 VPC 端点。
- 某些容器软件产品使用访问 AWS Marketplace Metering Service 的 API 调用来监控用量。私有集群不允许这些调用，因此您不能在私有集群中使用这些容器类型。

Amazon EKS Kubernetes 版本

Kubernetes 发展迅速，提供了新功能、设计更新和错误修复。社区平均每四个月发布一次新的 Kubernetes 次要版本（例如 `1.30`）。Amazon EKS 会遵循次要版本的上游发布和弃用周期。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。

在发布后的前 14 个月内，次要版本会在 Amazon EKS 中获得标准支持。一旦某个版本超过标准支持结束日期，它将在接下来 12 个月自动进入延期支持。延期支持允许您在特定 Kubernetes 版本上停留更长时间，但每个集群小时需支付额外费用。如果您在延期支持期限结束之前未更新集群，则您的集群将自动升级到当前支持的最旧延期版本。

我们建议您使用 Amazon EKS 支持的最新可用 Kubernetes 版本创建集群。如果您的应用程序需要特定版本的 Kubernetes，则可以选择较旧的版本。您可以在标准或延期支持中提供的任何版本上创建新的 Amazon EKS 集群。

标准支持上的可用版本

Amazon EKS 标准支持目前提供以下 Kubernetes 版本：

- 1.30
- 1.29
- 1.28
- 1.27
- 1.26

有关标准支持中每个版本需要注意的重要更改，请参阅 标准支持版本的发布说明。

延期支持的可用版本

Amazon EKS 扩展支持目前提供以下 Kubernetes 版本：

- 1.25
- 1.24
- 1.23

有关扩展支持中每个版本需要注意的重要更改，请参阅 扩展支持版本的发布说明。

Amazon EKS 扩展支持目前提供以下 Kubernetes 版本，附加要求是您不能使用这些版本创建新集群：

- 1.22
- 1.21

  有关这些版本的信息，请参阅 版本 1.21 和 1.22 的发行说明

Amazon EKS Kubernetes 发布日历

下表显示了每个 Kubernetes 版本需要考虑的重要发布和支持日期。

> **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	标准支持结束日期	延期支持结束日期
1.30	2024 年 4 月 17 日	2024 年 5 月 23 日	2025 年 7 月 23 日	2026 年 7 月 23 日
1.29	2023 年 12 月 13 日	2024 年 1 月 23 日	2025 年 3 月 23 日	2026 年 3 月 23 日
1.28	2023 年 8 月 15 日	2023 年 9 月 26 日	2024 年 11 月 26 日	2025 年 11 月 26 日
1.27	2023 年 4 月 11 日	2023 年 5 月 24 日	2024 年 7 月 24 日	2025 年 7 月 24 日
1.26	2022 年 12 月 9 日	2023 年 4 月 11 日	2024 年 6 月 11 日	2025 年 6 月 11 日
1.25	2022 年 8 月 23 日	2023 年 2 月 22 日	2024 年 5 月 1 日	2025 年 5 月 1 日
1.24	2022 年 5 月 3 日	2022 年 11 月 15 日	2024 年 1 月 31 日	2025 年 1 月 31 日
1.23	2021 年 12 月 7 日	2022 年 8 月 11 日	2023 年 10 月 11 日	2024 年 10 月 11 日
1.22	2021 年 8 月 4 日	2022 年 4 月 22 日	2023 年 6 月 4 日	2024 年 9 月 1 日
1.21	2021 年 4 月 8 日	2021 年 7 月 19 日	2023 年 2 月 16 日	2024 年 7 月 15 日

Amazon EKS 版本常见问题

标准支持提供多少个 Kubernetes 版本？

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为至少四个生产就绪的 Kubernetes 版本提供标准支持。我们会提前至少 60 天公告特定 Kubernetes 次要版本的标准支持结束日期。鉴于新 Kubernetes 版本的 Amazon EKS 鉴定和发布流程，Amazon EKS 上对某个 Kubernetes 版本的标准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。

Kubernetes 可以获得多长时间的 Amazon EKS 标准支持？

一个 Kubernetes 版本首次在 Amazon EKS 上发布后在 14 个月内获得标准支持。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。

当对 Amazon EKS 上的 Kubernetes 版本的标准支持结束时，我是否会收到通知？

是。如果您账户中有任何集群正在运行即将终止支持的版本，Amazon EKS 会在 Kubernetes 版本在 Amazon EKS 上发布大约 12 个月后通过 AWS Health Dashboard 发出通知。该通知包括支持结束日期。为从通知发出之日起至少 60 天。

Amazon EKS 支持哪些 Kubernetes 功能？

Amazon EKS 支持 Kubernetes API 的所有正式发布（GA）功能。从 Kubernetes 版本 `1.24` 开始，默认情况下，集群中不会启用新的测试版 API。但在默认情况下，先前存在的测试版 API 和现有测试版 API 的新版本将继续默认启用。不支持 Alpha 功能。

Amazon EKS 托管节点组是否会随集群控制面板版本一起自动更新？

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。有关更多信息，请参阅 更新托管节点组。我们建议您的控制面板和节点上保持相同的 Kubernetes 版本。

自行管理的节点组是否会随集群控制面板版本一起自动更新？

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 自行管理节点的更新。

Kubernetes 项目最多测试三个次要版本的控制面板和节点之间的兼容性。例如，当 `1.27` 节点由 `1.30` 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三个次要

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。

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？

不会。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。然后依次重启所有的 Fargate Pods。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。有关更多信息，请参阅 Kubernetes 文档中的部署。

> **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 扩展支持常见问题

标准支持和延期支持术语对我来说是全新的。这些术语是什么意思？

对 Amazon EKS 中某个 Kubernetes 版本的标准支持从该 Kubernetes 版本在 Amazon EKS 上发布时开始，并将在发布日期 14 个月后结束。对某个 Kubernetes 版本的延期支持将在标准支持结束后立即开始，并将在接下来的 12 个月后结束。例如，Amazon EKS 版本 `1.23` 的标准支持将于 2023 年 10 月 11 日结束。对版本 `1.23` 的延期支持于 2023 年 10 月 12 日开始，并将于 2024 年 10 月 11 日结束。

我需要做些什么才能获得对 Amazon EKS 集群的延期支持？

您无需采取任何措施即可获得对 Amazon EKS 集群的延期支持。标准支持从 Kubernetes 版本在 Amazon EKS 上发布时开始，并将在发布日期 14 个月后结束。对某个 Kubernetes 版本的延期支持将在标准支持结束后立即开始，并将在接下来的 12 个月后结束。在标准支持结束后的 Kubernetes 版本上运行的集群将自动加入延期支持。

哪些 Kubernetes 版本可以获得延期支持？

延期支持适用于 Kubernetes 版本 `1.23` 及更高版本。在任何版本的标准支持结束后，您最多可以在该版本上运行集群 12 个月。这意味着 Amazon EKS 将为每个版本提供 26 个月的支持（14 个月的标准支持加上 12 个月的延期支持）。

如果我不想使用延期支持怎么办？

如果您不想自动注册延期支持，则可以将集群升级到标准 Amazon EKS 支持的 Kubernetes 版本。未升级到标准支持 Kubernetes 版本的集群将自动进入延期支持。

12 个月的延期支持结束后会发生什么？

在已完成 26 个月生命周期（14 个月的标准支持加上 12 个月的延期支持）的 Kubernetes 版本上运行的集群将自动升级到下一个版本。

在延期支持结束之日，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 更新 Amazon EKS 集群的 Kubernetes 版本。

在延期支持结束日期后，自动更新控制面板的确切时间是何时？

Amazon EKS 无法给出具体的时间范围。自动更新可以在延期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 更新 Amazon EKS 集群 Kubernetes 版本。

我能否无限期地将控制面板留在某个 Kubernetes 版本上？

不能。AWS 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延期支持已经结束的版本上。

获得延期支持是否需要额外费用？

需要，在延期支持下运行的 Amazon EKS 集群需要支付额外费用。有关定价详情，请参阅 AWS 博客上的对 Kubernetes 版本定价的 Amazon EKS 扩展支持。

延期支持中包含哪些内容？

延期支持中的 Amazon EKS 集群会一直收到 Kubernetes 控制面板的安全补丁。此外，Amazon EKS 将发布 Amazon VPC CNI 的补丁 `kube-proxy` 和延期支持版本的 CoreDNS 附加组件。Amazon EKS 还将针对 AWS 发布的适用于 Amazon Linux、Bottlerocket 和 Windows 的 Amazon EKS 优化版 AMI 以及这些版本的 Amazon EKS Fargate 节点发布补丁。延期支持中的所有集群将继续从 AWS 中获得技术支持。

> **Note**
>
> AWS 发布的 Amazon EKS 优化版 Windows AMI 的延期支持不适用于 Kubernetes 版本 1.23，但适用于 Kubernetes 版本 1.24 及更高版本。

在延期支持中，非 Kubernetes 组件的补丁是否有任何限制？

虽然延期支持涵盖了来自 AWS 的所有 Kubernetes 特定组件，但它仅在任何时候为 AWS 发布的适用于 Amazon Linux、Bottlerocket 和 Windows 的 Amazon EKS 优化版 AMI 提供支持。这意味着，在使用延期支持时，您的 Amazon EKS 优化版 AMI 上可能会有更新的组件（例如操作系统或内核）。例如，一旦 Amazon Linux 2 在 2025 年结束其生命周期后，Amazon EKS 优化版 Amazon Linux AMI 将使用更新的 Amazon Linux 操作系统构建。Amazon EKS 将宣布并记录重要的支持生命周期差异，例如每个 Kubernetes 版本的差异。

我是否能使用扩展支持的版本创建新集群？

是，但 1.22 和 1.21 除外。例如，您可以创建 1.23 集群，但不能创建 1.22 集群。

标准支持版本的发布说明

本主题介绍有关标准支持中每个 Kubernetes 版本需要注意的重要更改。升级时，请仔细检查集群新旧版本之间发生的变化。

> **Note**
>
> 对于 1.24 和更高版本的集群，正式发布的 Amazon EKS AMI 包括 containerd 作为唯一的运行时。低于 1.24 的 Kubernetes 版本将使用 Docker 作为默认运行时。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 结束了对 Dockershim 的支持。

Kubernetes 1.30

Kubernetes 1.30 现已在 Amazon EKS 中推出。有关 Kubernetes 1.30 的更多信息，请参阅官方发布公告。

> **Important**
>
> - 从 Amazon EKS 版本 `1.30` 或更高版本开始，任何新创建的托管节点组都将自动默认使用 Amazon Linux 2023（AL2023）作为节点操作系统。以前，新节点组将默认为 Amazon Linux 2（AL2）。在创建新节点组时，您可以通过选择 AL2 作为 AMI 类型来继续使用 AL2。
>   - 有关 Amazon Linux 的更多信息，请参阅 Amazon Linux 用户指南中的比较 AL2 和 AL2023。
>   - 有关为托管节点组指定操作系统的更多信息，请参阅 创建托管节点组。

- 使用 Amazon EKS `1.30`，将 `topology.k8s.aws/zone-id` 标签添加到工作线程节点中。您可以使用可用区 ID（AZ ID），以确定一个账户中的资源相对于另一个账户中的资源所在的位置。有关更多信息，请参阅 AWS RAM 用户指南中的适用于 AWS 资源的可用区 ID。
- 从 `1.30` 开始，Amazon EKS 不再包含有关应用于新创建集群的 `gp2StorageClass` 资源的 `default` 注释。如果您按名称引用此存储类，则不会产生任何影响。如果您依赖集群中具有默认 `StorageClass`，则必须采取措施。您应该通过名称 `StorageClass` 来引用 `gp2`。或者，您可以通过在安装 `defaultStorageClass.enabled` 的 `v1.31.0` 或更高版本时将 `aws-ebs-csi-driver add-on` 参数设置为 true 来部署 Amazon EBS 建议的默认存储类。
- Amazon EKS 集群 IAM 角色所需的最低 IAM 策略已更改。操作 `ec2:DescribeAvailabilityZones` 是必需的。有关更多信息，请参阅 Amazon EKS 集群 IAM 角色。

有关完整的 Kubernetes `1.30` 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.30.md。

Kubernetes 1.29

Kubernetes `1.29` 现已在 Amazon EKS 中推出。有关 Kubernetes `1.29` 的更多信息，请参阅官方发布公告。

> **Important**
>
> - Kubernetes v1.29 中不再提供 FlowSchema 和 PriorityLevelConfiguration 已弃用的 flowcontrol.apiserver.k8s.io/v1beta2 API 版本。如果您有使用已弃用 beta API 组的对象或客户端软件，则应该先更改这些，然后再升级到 v1.29。

- 节点对象的 .status.kubeProxyVersion 字段已弃用，而 Kubernetes 项目在未来版本中延迟删除此字段。已弃用的字段不正确，且过去一直由 kubelet 管理 - 并不真的知道 kube-proxy 版本，或 kube-proxy 是否正在运行。如果您一直在客户端软件中使用此字段，请停止使用 - 其中的信息不可靠且该字段已弃用。
- 在 Kubernetes 1.29 中用于减少潜在攻击面，LegacyServiceAccountTokenCleanUp 功能将自动生成的旧基于密钥的令牌标记为无效，如果这些令牌长时间未使用的话（默认为 1 年），且如果在标记为无效后长时间无人使用，则会自动删除（默认为额外增加 1 年）。可运行以下内容识别此类令牌：

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -nkube-system
```

有关完整的 Kubernetes 1.29 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#changelog-since-v1280。

Kubernetes 1.28

Kubernetes 1.28 现已在 Amazon EKS 中推出。有关 Kubernetes 1.28 的更多信息，请参阅官方发布公告。

- Kubernetes v1.28 将核心节点和控制面板组件之间支持的偏差扩大了一个次要版本，从 n-2 到 n-3，这样支持的最旧次要版本的节点组件（kubelet 和 kube-proxy）便可与支持的最新次要版本的控制面板组件（kube-apiserver、kube-scheduler、kube-controller-manager、cloud-controller-manager）一起使用。
- Pod GC Controller 中的指标 force_delete_pods_total 和 force_delete_pod_errors_total 已增强，可以将所有强制容器组（pod）删除考虑在内。在指标中添加了一个原因，指示容器组（pod）是由于已终止、已孤立、因服务中断污点而终止还是已终止但未计划。
- PersistentVolume（PV）控制器已修改为自动为任何 storageClassName 未设置的未绑定 PersistentVolumeClaim 分配默认值 StorageClass。此外，API 服务器内的

PersistentVolumeClaim 准入验证机制已经过调整，允许将值从未设置状态更改为实际 StorageClass 名称。

有关完整的 Kubernetes 1.28 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.28.md#changelog-since-v1270。

Kubernetes 1.27

Kubernetes 1.27 现已在 Amazon EKS 中推出。有关 Kubernetes 1.27 的更多信息，请参阅官方发布公告。

Important

- 已移除对 alpha seccomp 注释 seccomp.security.alpha.kubernetes.io/pod 和 container.seccomp.security.alpha.kubernetes.io 注释的支持。alpha seccomp 注释在 1.19 中已弃用，随着它们在 1.27 中的移除，Pods 的 seccomp 字段将不再自动填充 seccomp 注释。反之，将会使用 Pods 或容器的 securityContext.seccompProfile 字段配置 seccomp 配置文件 要检查您是否在集群中使用了弃用的 alpha seccomp 注释，请运行以下命令：

  ```
  kubectl get pods --all-namespaces -o json | grep
   -E 'seccomp.security.alpha.kubernetes.io/pod|
  container.seccomp.security.alpha.kubernetes.io'
  ```

- kubelet 的 --container-runtime 命令行参数已移除。Amazon EKS 的默认容器运行时系统自 1.24 开始便已 containerd，从而不需要指定容器运行时系统。从 1.27 开始，Amazon EKS 将忽略传递给任何引导脚本的 --container-runtime 参数。重要的是不要将此参数传递给 --kubelet-extra-args，以防止在节点引导过程中出现错误。您必须从所有节点创建工作流和生成脚本中删除 --container-runtime 参数。

- Kubernetes 1.27 中的 kubelet 将默认值 kubeAPIQPS 增加到 50，将 kubeAPIBurst 增加到 100。这些增强功能使 kubelet 能够处理更多的 API 查询，从而缩短响应时间并提高性能。当 Pods 的需求由于扩展要求而增加时，修订后的默认值可确保 kubelet 能够有效地管理增加的工作负载。因此，Pod 启动速度更快，集群操作更有效。
- 您可以使用更精细的 Pod 拓扑结构来传播策略，例如 minDomain。此参数使您能够指定 Pods 应分布的最小域数量。nodeAffinityPolicy 和 nodeTaintPolicy 允许在管理 Pod 分布时提供

额外的精细度。这与您 Pod's 规范的 topologySpreadConstraints 中的节点亲和性、污点和 matchLabelKeys 字段一致。这样一来，便可以在滚动升级后选择 Pods 进行发散计算。

- Kubernetes1.27 已升级到测试版，StatefulSets 的控制其 PersistentVolumeClaims（PVCs）生命周期的新策略机制。新的 PVC 保留策略可让您指定在删除 StatefulSet 或缩减 StatefulSet 中的副本时，通过 StatefulSet 规范模板生成的 PVCs 将会自动删除还是保留。
- Kubernetes API 服务器中的 goaway-chance 选项通过随机关闭连接，帮助防止 HTTP/2 客户端连接卡在单个 API 服务器实例上。连接关闭后，客户端将尝试重新连接，并且由于负载均衡，很可能会登录不同的 API 服务器。Amazon EKS 版本 1.27 已启用 goaway-chance 标志。如果您在 Amazon EKS 集群上运行的工作负载使用的客户端与 HTTP GOAWAY 不兼容，则建议您在连接终止时重新连接，以更新您的客户端处理 GOAWAY。

有关完整的 Kubernetes 1.27 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.27.md#changelog-since-v1260。

Kubernetes 1.26

Kubernetes 1.26 现已在 Amazon EKS 中推出。有关 Kubernetes 1.26 的更多信息，请参阅官方发布公告。

> **Important**
>
> Kubernetes 1.26 不再支持 CRI v1alpha2。如果容器运行时系统不支持 CRI v1，这会导致 kubelet 不再注册该节点。这也意味着 Kubernetes 1.26 不支持 containerd 1.5 次要版本及更早版本。如果您使用的是 containerd，则需要先升级到 containerd 1.6.0 版本或更高版本，然后再将任何节点升级到 Kubernetes 1.26。您还需要升级任何其他仅支持 v1alpha2 的容器运行时系统。有关更多信息，请咨询容器运行时系统供应商。默认情况下，Amazon Linux 和 Bottlerocket AMI 包含 containerd 1.6.6 版本。

- 在升级到 Kubernetes 1.26 之前，请将您的 Amazon VPC CNI plugin for Kubernetes 升级到 1.12 版本或更高版本。如果您不升级到 Amazon VPC CNI plugin for Kubernetes 版本 1.12 或者更高版本，则 Amazon VPC CNI plugin for Kubernetes 将会崩溃。有关更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。
- Kubernetes API 服务器中的 goaway-chance 选项通过随机关闭连接，帮助防止 HTTP/2 客户端连接卡在单个 API 服务器实例上。连接关闭后，客户端将尝试重新连接，并且由于负载均衡，很可

能会登录不同的 API 服务器。Amazon EKS 版本 1.26 已启用 goaway-chance 标志。如果您在 Amazon EKS 集群上运行的工作负载使用的客户端与 HTTP GOAWAY 不兼容，则建议您在连接终止时重新连接，以更新您的客户端处理 GOAWAY。

有关完整的 Kubernetes 1.26 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.26.md#changelog-since-v1250。

扩展支持版本的发布说明

本主题介绍有关扩展支持中每个 Kubernetes 版本需要注意的重要更改。升级时，请仔细检查集群新旧版本之间发生的变化。

Kubernetes 1.25

Kubernetes 1.25 现已在 Amazon EKS 中推出。有关 Kubernetes 1.25 的更多信息，请参阅官方发布公告。

Important

- 从 Kubernetes 版本 1.25 开始，您将无法再将 Amazon EC2 P2 实例与现成的 Amazon EKS 优化加速 Amazon Linux AMI 结合使用。对于 Kubernetes 版本 1.25 或更高版本，这些 AMI 将支持 NVIDIA 525 系列或更高版本的驱动程序，这些驱动程序与 P2 实例不兼容。但 NVIDIA 525 系列或更高版本的驱动程序与 P3、P4 和 P5 实例兼容，因此您可以通过 Kubernetes 版本 1.25 或更高版本的 AMI 使用这些实例。请首先将所有 P2 实例迁移到 P3、P4 和 P5 实例，然后再将您的 Amazon EKS 集群升级到版本 1.25。您还应主动升级您的应用程序以支持 NVIDIA 525 系列或更高版本。我们计划在 2024 年 1 月晚些时候将更新的 NVIDIA 525 系列或更新驱动程度反向移植到 Kubernetes 版本 1.23 和 1.24。
- PodSecurityPolicy（PSP）已在 Kubernetes 1.25 中删除。PSPs 被容器组（pod）安全准入（PSA）和容器组（pod）安全标准 (PSS) 替代。PSA 是一个内置的准入控制器，它实施 PSS 中所述的安全控件。PSA 和 PSS 在 Kubernetes 1.25 中逐步达到稳定，并在 Amazon EKS 中被默认启用。如果您的集群中有 PSPs，则请确保在将集群升级到版本 1.25 之前，从 PSP 迁移到内置的 Kubernetes PSS 或策略即代码解决方案。如果您不从 PSP 迁移，则您的工作负载可能会中断。有关更多信息，请参阅 容器组（pod）安全策略 (PSP) 移除常见问题。
- Kubernetes 版本 1.25 包含了若干更改，这些更改会修改名为 API 优先级和公平性（APF）的现有功能的行为。在出现大量的请求期间，APF 可保护 API 服务器免受可能的过载影响。为此，它会限制在任何给定时间可以处理的并发请求数量。实现这一点的方法是对来自不同

工作负载或用户的请求执行不同的优先级和限制。这种方法可确保优先处理关键应用程序或高优先级请求，同时防止低优先级请求使 API 服务器不堪重负。有关更多信息，请参见 Kubernetes 文档中的 API 优先级和公平性 或《EKS 最佳实践指南》中的 API 优先级和公平性。

这些更新是在 PR #10352 和 PR #118601 中推出的。以前，APF 对所有类型的请求会一视同仁，每个请求都会消耗并发请求限制的一个单位。APF 行为更改后，系统会将较高的并发单位分配给 LIST 请求，因为这些请求给 API 服务器带来了异常沉重的负担。API 服务器会估算 LIST 请求将要返回的对象数量。然后按照返回的对象数量分配一个相适应的并发单位。

升级到 Amazon EKS 版本 1.25 或更高版本后，此更新后的行为可能会导致有大量 LIST 请求工作负载（以前可以正常运行）受到速率限制。这将通过 HTTP 429 响应代码来提示。为避免因 LIST 请求受到速率限制而可能导致的工作负载中断，我们强烈建议您调整工作负载以降低此类请求的速率。您还可以通过调整 APF 设置，为关键请求分配更多容量，同时减少分配给非关键请求的容量，从而解决此问题。有关这些问题缓解技术的更多信息，请参阅《EKS 最佳实践指南》中的 防止请求被丢弃。

- Amazon EKS 1.25 包括对集群身份验证的增强，其中包含更新的 YAML 库。如果在 kube-system 命名空间中找到的 aws-auth ConfigMap 的 YAML 值以宏开头，则其中第一个字符是大括号，则应在大括号（{ }）之前和之后添加引号（“ ”）。确保 aws-iam-authenticator 版本 v0.6.3 准确解析 Amazon EKS 1.25 中的 aws-auth ConfigMap 需要此操作。
- EndpointSlice 的测试版 API（discovery.k8s.io/v1beta1）已在 Kubernetes 1.21 中弃用，自 Kubernetes 1.25 起不再提供。此 API 已更新为 discovery.k8s.io/v1。有关更多信息，请参阅 Kubernetes文档中的 EndpointSlice。AWS Load Balancer Controller v2.4.6 和更早版本使用 v1beta1 端点与 EndpointSlices 通信。如果您将 EndpointSlices 配置用于 AWS Load Balancer Controller，则必须先升级到 AWS Load Balancer Controller v2.4.7，然后才能将 Amazon EKS 集群升级到 1.25。如果您在将 EndpointSlices 配置用于 AWS Load Balancer Controller 时升级到 1.25，则控制器将崩溃并导致您的工作负载中断。要升级控制器，请参阅 AWS Load Balancer Controller 是什么？。

- SeccompDefault 已提升到 Kubernetes 1.25 中的测试版。通过在配置 kubelet 时设置 --seccomp-default 标志，容器运行时使用其 RuntimeDefaultseccomp 配置文件，而不是无约束（seccomp disabled）模式。默认配置文件提供了一组强大的安全默认值，同时保留了工作负载的功能。尽管此标志可用，但默认情况下，Amazon EKS 不启用此标志，因此 Amazon EKS 的行

为实际上没有变化。如果愿意，您可以开始在您的节点上启用这个功能。有关更多详细信息，请参阅 Kubernetes 文档中的教程使用 seccomp 限制容器的系统调用。

- Kubernetes 1.24 及更高版本中删除了对 Docker（也称为 Dockershim）的容器运行时接口（CRI）的支持。Kubernetes 1.24 及更高版本集群的 Amazon EKS 官方 AMIs 的唯一容器运行时是 containerd。在升级到 Amazon EKS 1.24 或更高版本之前，删除对不再支持的引导脚本标志的任何引用。有关更多信息，请参阅 Amazon EKS 结束了对 Dockershim 的支持。
- 对通配符查询的支持已在 CoreDNS 1.8.7 中被弃用并在 CoreDNS 1.9 中被移除。此操作作为一项安全措施而执行。通配符查询不再起作用，并返回 NXDOMAIN 而不是 IP 地址。
- Kubernetes API 服务器中的 goaway-chance 选项通过随机关闭连接，帮助防止 HTTP/2 客户端连接卡在单个 API 服务器实例上。连接关闭后，客户端将尝试重新连接，并且由于负载均衡，很可能会登录不同的 API 服务器。Amazon EKS 版本 1.25 已启用 goaway-chance 标志。如果您在 Amazon EKS 集群上运行的工作负载使用的客户端与 HTTP GOAWAY 不兼容，则建议您在连接终止时重新连接，以更新您的客户端处理 GOAWAY。

有关完整的 Kubernetes 1.25 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.25.md#changelog-since-v1240。

Kubernetes 1.24

Kubernetes 1.24 现已在 Amazon EKS 中推出。有关 Kubernetes 1.24 的更多信息，请参阅官方发布公告。

Important

- 从 Kubernetes 1.24 开始，默认情况下集群中不启用新的测试版 API。默认情况下，现有的测试版 API 及其新版本继续处于启用状态。Amazon EKS 遵循与上游的 Kubernetes 1.24 相同的行为。默认情况下，控制新 API 操作和现有 API 操作的新功能的功能门控处于启用状态。这与上游 Kubernetes 一致。有关更多信息，请参阅 GitHub 上的 KEP-3136: Beta APIs Are Off by Default（KEP-3136：测试版 API 默认处于关闭状态）。
- Kubernetes 1.24 中删除了对 Docker（也称为 Dockershim）的容器运行时接口（CRI）的支持。Amazon EKS 官方 AMI 将 containerd 作为唯一的运行时。在迁移到 Amazon EKS 1.24 或更高版本之前，必须删除对不再支持的引导脚本标志的任何引用。您还必须确保已为 Worker 节点启用 IP 转发。有关更多信息，请参阅 Amazon EKS 结束了对 Dockershim 的支持。
- 如果您已经为 Container Insights 进行了 Fluentd 配置，则必须先将 Fluentd 迁移到 Fluent Bit，然后才能更新集群。Fluentd 解析器配置为仅解析 JSON 格式的日志消息。与 dockerd

不同的是，containerd 容器运行时系统的日志消息不是 JSON 格式的。如果您不迁移到 Fluent Bit，一些配置的 Fluentd's 解析器将在 Fluentd 容器内生成大量错误。有关迁移的更多信息，请参阅将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs。

- 在 Kubernetes 1.23 及早期版本中，具有不可验证的 IP 和 DNS 使用者备用名称（SAN）的 kubelet 服务证书会自动使用不可验证的 SAN 进行颁发。预置的证书中省略了这些不可验证的 SAN。在 1.24 版及更高版本的集群中，如果无法验证任何 SAN，则不会颁发 kubelet 服务证书。这会阻止 kubectl exec 和 kubectl logs 命令发挥作用。有关更多信息，请参阅 将集群升级到 Kubernetes 1.24 之前的证书签名注意事项。
- 升级使用 Fluent Bit 的 Amazon EKS 1.23 集群时，必须确保其运行 k8s/1.3.12 或更高版本。要实现这一点，您可以重新应用来自 GitHub 的最新适用 Fluent Bit YAML 文件。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的设置 Fluent Bit。

- 当跨多个可用区部署集群 Worker 节点时，您可以使用拓扑感知提示（Topology Aware Hints）来表明您希望将流量保持在区域内。在区域内路由流量有助于降低成本和提高网络性能。默认情况下，拓扑感知提示在 Amazon EKS 1.24 中处于启用状态。有关更多信息，请参阅 Kubernetes 文档中的 Topology Aware Hints（拓扑感知提示）。
- Kubernetes 1.25 计划删除 PodSecurityPolicy（PSP）。PSPs 正在被 Pod Security Admission (PSA)（容器组（pod）安全准入（PSA））所取代。PSA 是一个内置的准入控制器，使用了 Pod Security Standards（PSS）（容器组（pod）安全标准（PSS））中所述的安全控件。PSA 和 PSS 都是测试版功能，默认情况下在 Amazon EKS 中处于启用状态。为了解决 1.25 版中的 PSP 删除问题，我们建议您在 Amazon EKS 中实施 PSS。有关更多信息，请参阅 AWS 博客上的 Implementing Pod Security Standards in Amazon EKS（在 Amazon EKS 中实施容器组（pod）安全标准）。
- Kubernetes 1.24 中删除了 client.authentication.k8s.io/v1alpha1 ExecCredential。ExecCredential API 通常在 Kubernetes 1.22 中可用。如果您使用依赖于 v1alpha1 API 的 client-go 凭证插件，请联系您的插件分销商，了解如何迁移到 v1 API。
- 对于 Kubernetes 1.24，我们为上游 Cluster Autoscaler 项目贡献了一项功能，该功能简化了从/向零节点扩展 Amazon EKS 托管节点组的过程。以前，要让 Cluster Autoscaler 了解扩展到零节点的托管节点组的资源、标签和污点，您需要使用其负责的节点的详细信息来标记底层 Amazon EC2 Auto Scaling 组。现在，当托管节点组中没有正在运行的节点时，Cluster Autoscaler 会调用 Amazon EKS DescribeNodegroup API 操作。此 API 操作提供了 Cluster Autoscaler 所需有关托管节点组的资源、标签和污点的信息。此功能要求您向 Cluster Autoscaler 服务账户 IAM 策略添加 eks:DescribeNodegroup 权限。当为 Amazon EKS 托管节点组提供支持的自动扩缩组上 Cluster

Autoscaler 标签的值与节点组本身发生冲突时，Cluster Autoscaler 倾向于使用自动扩缩组标签的值。这样您就能根据需要覆盖值。有关更多信息，请参阅 Autoscaling。

- 如果您打算在 Amazon EKS 1.24 中使用 Inferentia 或 Trainium 实例类型，则必须升级到 AWS Neuron 设备插件 1.9.3.0 版或更高版本。有关更多信息，请参阅 AWS Neuron 文档中的 Neuron K8 release [1.9.3.0]（Neuron K8 版本 [1.9.3.0]）。
- 默认情况下，Containerd 已为 Pods 启用 IPv6。它将节点内核设置应用于 Pod 网络命名空间。因此，Pod 中的容器绑定到 IPv4（127.0.0.1）和 IPv6（::1）环回地址。IPv6 是通信的默认协议。在将集群更新到版本 1.24 之前，建议您测试您的多容器 Pods。修改应用程序，使其可以绑定到环回接口上的所有 IP 地址。大多数库都启用了 IPv6 绑定，绑定向后兼容 IPv4。当无法修改您的应用程序代码时，您有两种选择：
  - 运行 init 容器并将 disable ipv6 设置为 true（sysctl -w net.ipv6.conf.all.disable_ipv6=1）。
  - 配置一个转换准入 Webhook 以在您的应用程序 Pods 旁注入 init 容器。

  如果您需要为所有节点上的所有 Pods 阻止 IPv6，则可能必须在实例上禁用 IPv6。
- Kubernetes API 服务器中的 goaway-chance 选项通过随机关闭连接，帮助防止 HTTP/2 客户端连接卡在单个 API 服务器实例上。连接关闭后，客户端将尝试重新连接，并且由于负载均衡，很可能会登录不同的 API 服务器。Amazon EKS 版本 1.24 已启用 goaway-chance 标志。如果您在 Amazon EKS 集群上运行的工作负载使用的客户端与 HTTP GOAWAY 不兼容，则建议您在连接终止时重新连接，以更新您的客户端处理 GOAWAY。

有关完整的 Kubernetes 1.24 更改日志，请参阅https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.24.md#changelog-since-v1230。

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 1.23 版之前，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。如果您计划在新的 1.23 版集群中部署使用 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 Amazon EBS CSI 驱动程序。有关迁移功能的常见问题，请参阅 Amazon EBS CSI 迁移常见问题。

- AWS 发布的 Amazon EKS 优化版 Windows AMI 的延期支持不适用于 Kubernetes 版本 1.23，但适用于 Kubernetes 版本 1.24 及更高版本。

- 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 正在舍弃 Dockershim：承诺和后续步骤。从 Amazon EKS 1.24 版开始，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 1.23 版继续支持 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 Amazon EKS 结束了对 Dockershim 的支持。
- 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（PSA）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]。PSA 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。当 `kubectl exec` 因容器崩溃或容器映像不包含调试实用程序而不足时，这种容器对于交互式排查问题尤其有用。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。`HorizontalPodAutoscaler autoscaling/v2beta2` API 已弃用。其在 `1.26` 中不可用。
- Kubernetes API 服务器中的 `goaway-chance` 选项通过随机关闭连接，帮助防止 HTTP/2 客户端连接卡在单个 API 服务器实例上。连接关闭后，客户端将尝试重新连接，并且由于负载均衡，很可能会登录不同的 API 服务器。Amazon EKS 版本 `1.23` 已启用 `goaway-chance` 标志。如果您在 Amazon EKS 集群上运行的工作负载使用的客户端与 `HTTP GOAWAY` 不兼容，则建议您在连接终止时重新连接，以更新您的客户端处理 `GOAWAY`。

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

版本 1.21 和 1.22 的发行说明

> **Important**
>
> 您无法使用这些版本创建新集群。

本主题介绍版本 `1.22` 和 `1.21` 需要注意的重要更改。升级时，请仔细检查集群新旧版本之间发生的变化。

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`、`MutatingAdmissionW`

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
`1.22.17`	`eks.28`	新的平台版本（包括安全修复和增强功能）。	2024 年 5 月 16 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.22.17	eks.26	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 1 日
1.22.17	eks.14	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 30 日
1.22.17	eks.13	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日
1.22.17	eks.12	新的平台版本（包括安全修复和增强功能）。	2023 年 5 月 5 日
1.22.17	eks.11	新的平台版本（包括安全修复和增强功能）。	2023 年 3 月 24 日
1.22.16	eks.10	新的平台版本（包括安全修复和增强功能）。	2023 年 1 月 27 日
1.22.15	eks.9	新的平台版本（包括安全修复和增强功能）。	2022 年 12 月 5 日
1.22.15	eks.8	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 18 日
1.22.15	eks.7	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 7 日
1.22.13	eks.6	新的平台版本（包括安全修复和增强功能）。	2022 年 9 月 21 日
1.22.10	eks.5	具有改进 etcd 弹性的新平台版本。	2022 年 8 月 15 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
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。	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。	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、MutatingAdmissionW

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.21.14	eks.33	新的平台版本（包括安全修复和增强功能）。	2024 年 5 月 16 日
1.21.14	eks.31	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 1 日
1.21.14	eks.18	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.21.14	eks.17	新的平台版本（包括安全修复和增强功能）。	2023 年 5 月 5 日
1.21.14	eks.16	新的平台版本（包括安全修复和增强功能）。	2023 年 3 月 24 日
1.21.14	eks.15	新的平台版本（包括安全修复和增强功能）。	2023 年 1 月 27 日
1.21.14	eks.14	新的平台版本（包括安全修复和增强功能）。	2022 年 12 月 5 日
1.21.14	eks.13	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 18 日
1.21.14	eks.12	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 7 日
1.21.13	eks.11	具有改进 etcd 弹性的新平台版本。	2022 年 10 月 10 日
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。	2022 年 7 月 21 日
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

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
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	AWS 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，请参阅为服务账户配置 AWS Security Token Service 端点。	2022 年 4 月 8 日
1.21.5	eks.5	使用 服务账户的 IAM 角色 时，现在将默认使用 AWS 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。	2021 年 12 月 13 日
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。	2021 年 11 月 8 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.21.2	eks.2	新的平台版本（包括安全修复和增强功能）。	2021 年 9 月 17 日
1.21.2	eks.1	面向 Amazon EKS 的 Kubernetes 版本 1.21 的初始版本。	2021 年 7 月 19 日

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。您可以使用 AWS CLI 或 AWS Management Console 检索集群的当前平台版本。如果您在 AWS Outposts 上有一个本地集群，请参阅 Amazon EKS 本地集群平台版本 而不是本主题。

当 Amazon EKS 中有新的 Kubernetes 次要版本可用时，例如 1.30，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 Amazon EKS 平台版本比当前平台版本落后两个版本以上。
- 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 1.30

所有 1.30 平台版本都启用了以下准入控制器：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.30.1	eks.3	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.30.0	eks.2	面向 EKS 的 Kubernetes 1.30 的初始版本。有关更多信息，请参阅 Kubernetes 1.30。	2024 年 5 月 23 日

Kubernetes 版本 1.29

所有 1.29 平台版本都启用了以下准入控制器：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.29.5	eks.8	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.29.4	eks.7	具有 CoreDNS 自动扩缩、安全修复和增强功能的新平台版本。有关 CoreDNS 自动扩缩的更多信息，请参阅 自动扩缩 CoreDNS。	2024 年 5 月 16 日
1.29.3	eks.6	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.29.1	eks.5	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.29.1	eks.4	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日
1.29.1	eks.3	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.29.0	eks.1	面向 EKS 的 Kubernetes 1.29 的初始版本。有关更多信息，请参阅 Kubernetes 1.29。	2024 年 1 月 23 日

Kubernetes 版本 **1.28**

所有 1.28 平台版本都启用了以下准入控制器：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.28.10	eks.14	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.28.9	eks.13	具有 CoreDNS 自动扩缩、安全修复和增强功能的新平台版本。有关 CoreDNS 自动扩缩的更多信息，请参阅 自动扩缩 CoreDNS。	2024 年 5 月 16 日
1.28.8	eks.12	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日
1.28.7	eks.11	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.28.7	eks.10	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.28.6	eks.9	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.28.5	eks.7	新的平台版本（包括安全修复和增强功能）。	2024 年 1 月 17 日
1.28.4	eks.6	具有 访问条目、安全修复和增强功能的新平台版本。	2023 年 12 月 14 日
1.28.4	eks.5	新的平台版本（包括安全修复和增强功能）。	2023 年 12 月 12 日
1.28.3	eks.4	具有 EKS 容器组身份、安全修复和增强功能的新平台版本。	2023 年 11 月 10 日
1.28.3	eks.3	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 3 日
1.28.2	eks.2	新的平台版本（包括安全修复和增强功能）。	2023 年 10 月 16 日
1.28.1	eks.1	面向 EKS 的 Kubernetes 1.28 的初始版本。有关更多信息，请参阅 Kubernetes 1.28。	2023 年 9 月 26 日

Kubernetes 版本 **1.27**

所有 1.27 平台版本都启用了以下准入控制器：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.27.14	eks.18	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.27.13	eks.17	具有 CoreDNS 自动扩缩、安全修复和增强功能的新平台版本。有关 CoreDNS 自动扩缩的更多信息，请参阅 自动扩缩 CoreDNS。	2024 年 5 月 16 日
1.27.12	eks.16	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日
1.27.11	eks.15	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.27.11	eks.14	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日
1.27.10	eks.13	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.27.9	eks.11	新的平台版本（包括安全修复和增强功能）。	2024 年 1 月 17 日
1.27.8	eks.10	具有 访问条目、安全修复和增强功能的新平台版本。	2023 年 12 月 14 日
1.27.8	eks.9	新的平台版本（包括安全修复和增强功能）。	2023 年 12 月 12 日
1.27.7	eks.8	具有 EKS 容器组身份、安全修复和增强功能的新平台版本。	2023 年 11 月 10 日
1.27.7	eks.7	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 3 日
1.27.6	eks.6	新的平台版本（包括安全修复和增强功能）。	2023 年 10 月 16 日
1.27.4	eks.5	新的平台版本（包括安全修复和增强功能）。	2023 年 8 月 30 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.27.4	eks.4	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 30 日
1.27.3	eks.3	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 30 日
1.27.2	eks.2	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日
1.27.1	eks.1	面向 EKS 的 Kubernetes 1.27 的初始版本。有关更多信息，请参阅 Kubernetes 1.27。	2023 年 5 月 24 日

Kubernetes 版本 **1.26**

所有 1.26 平台版本都启用了以下准入控制器：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.26.15	eks.19	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.26.15	eks.18	具有 CoreDNS 自动扩缩、安全修复和增强功能的新平台版本。有关 CoreDNS 自动扩缩的更多信息，请参阅 自动扩缩 CoreDNS。	2024 年 5 月 16 日
1.26.15	eks.17	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日
1.26.14	eks.16	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.26.14	eks.15	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.26.13	eks.14	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.26.12	eks.12	新的平台版本（包括安全修复和增强功能）。	2024 年 1 月 17 日
1.26.11	eks.11	具有访问条目、安全修复和增强功能的新平台版本。	2023 年 12 月 14 日
1.26.11	eks.10	新的平台版本（包括安全修复和增强功能）。	2023 年 12 月 12 日
1.26.10	eks.9	具有 EKS 容器组身份、安全修复和增强功能的新平台版本。	2023 年 11 月 10 日
1.26.10	eks.8	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 3 日
1.26.9	eks.7	新的平台版本（包括安全修复和增强功能）。	2023 年 10 月 16 日
1.26.7	eks.6	新的平台版本（包括安全修复和增强功能）。	2023 年 8 月 30 日
1.26.7	eks.5	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 30 日
1.26.6	eks.4	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 30 日
1.26.5	eks.3	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日
1.26.4	eks.2	新的平台版本（包括安全修复和增强功能）。	2023 年 5 月 5 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.26.2	eks.1	面向 EKS 的 Kubernetes 1.26 的初始版本。有关更多信息，请参阅 Kubernetes 1.26。	2023 年 4 月 11 日

Kubernetes 版本 1.25

所有 1.25 平台版本都启用了以下准入控制器：NodeRestriction、ExtendedResourceToleration、NamespaceLifecycle、LimitRanger、

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.25.16	eks.20	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.25.16	eks.19	具有 CoreDNS 自动扩缩、安全修复和增强功能的新平台版本。有关 CoreDNS 自动扩缩的更多信息，请参阅 自动扩缩 CoreDNS。	2024 年 5 月 16 日
1.25.16	eks.18	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日
1.25.16	eks.17	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.25.16	eks.16	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日
1.25.16	eks.15	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.25.16	eks.13	新的平台版本（包括安全修复和增强功能）。	2024 年 1 月 17 日
1.25.16	eks.12	具有 访问条目、安全修复和增强功能的新平台版本。	2023 年 12 月 14 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.25.16	eks.11	新的平台版本（包括安全修复和增强功能）。	2023 年 12 月 12 日
1.25.15	eks.10	具有 EKS 容器组身份、安全修复和增强功能的新平台版本。	2023 年 11 月 10 日
1.25.15	eks.9	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 3 日
1.25.14	eks.8	新的平台版本（包括安全修复和增强功能）。	2023 年 10 月 16 日
1.25.12	eks.7	新的平台版本（包括安全修复和增强功能）。	2023 年 8 月 30 日
1.25.12	eks.6	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 30 日
1.25.11	eks.5	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 30 日
1.25.10	eks.4	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日
1.25.9	eks.3	新的平台版本（包括安全修复和增强功能）。	2023 年 5 月 5 日
1.25.8	eks.2	新的平台版本（包括安全修复和增强功能）。	2023 年 3 月 24 日
1.25.6	eks.1	面向 EKS 的 Kubernetes 1.25 的初始版本。有关更多信息，请参阅 Kubernetes 1.25。	2023 年 2 月 21 日

Kubernetes 版本 1.24

所有 1.24 平台版本都启用了以下准入控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Defaul

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.24.17	eks.23	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.24.17	eks.22	新的平台版本（包括安全修复和增强功能）。	2024 年 5 月 16 日
1.24.17	eks.21	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日
1.24.17	eks.20	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.24.17	eks.19	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日
1.24.17	eks.18	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.24.17	eks.16	新的平台版本（包括安全修复和增强功能）。	2024 年 1 月 17 日
1.24.17	eks.15	具有 访问条目、安全修复和增强功能的新平台版本。	2023 年 12 月 14 日
1.24.17	eks.14	新的平台版本（包括安全修复和增强功能）。	2023 年 12 月 12 日
1.24.17	eks.13	具有 EKS 容器组身份、安全修复和增强功能的新平台版本。	2023 年 11 月 10 日
1.24.17	eks.12	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 3 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.24.17	eks.11	新的平台版本（包括安全修复和增强功能）。	2023 年 10 月 16 日
1.24.16	eks.10	新的平台版本（包括安全修复和增强功能）。	2023 年 8 月 30 日
1.24.16	eks.9	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 30 日
1.24.15	eks.8	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 30 日
1.24.14	eks.7	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日
1.24.13	eks.6	新的平台版本（包括安全修复和增强功能）。	2023 年 5 月 5 日
1.24.12	eks.5	新的平台版本（包括安全修复和增强功能）。	2023 年 3 月 24 日
1.24.8	eks.4	新的平台版本（包括安全修复和增强功能）。	2023 年 1 月 27 日
1.24.7	eks.3	新的平台版本（包括安全修复和增强功能）。	2022 年 12 月 5 日
1.24.7	eks.2	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 18 日
1.24.7	eks.1	面向 EKS 的 Kubernetes 1.24 的初始版本。有关更多信息，请参阅 Kubernetes 1.24。	2022 年 11 月 15 日

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、Defaul

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.23.17	eks.25	新的平台版本（包括安全修复和增强功能）。	2024 年 6 月 7 日
1.23.17	eks.24	新的平台版本（包括安全修复和增强功能）。	2024 年 5 月 16 日
1.23.17	eks.23	新的平台版本（包括安全修复和增强功能）。	2024 年 4 月 18 日
1.23.17	eks.22	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 29 日
1.23.17	eks.21	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 20 日
1.23.17	eks.20	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 12 日
1.23.17	eks.18	新的平台版本（包括安全修复和增强功能）。	2024 年 1 月 17 日
1.23.17	eks.17	具有 访问条目、安全修复和增强功能的新平台版本。	2023 年 12 月 14 日
1.23.17	eks.16	新的平台版本（包括安全修复和增强功能）。	2023 年 12 月 12 日
1.23.17	eks.15	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 10 日
1.23.17	eks.14	新的平台版本（包括安全修复和增强功能）。	2023 年 11 月 3 日

Kubernetes 版本	EKS 平台版本	发布说明	发行日期
1.23.17	eks.13	新的平台版本（包括安全修复和增强功能）。	2023 年 10 月 16 日
1.23.17	eks.12	新的平台版本（包括安全修复和增强功能）。	2023 年 8 月 30 日
1.23.17	eks.11	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 30 日
1.23.17	eks.10	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 30 日
1.23.17	eks.9	新的平台版本（包括安全修复和增强功能）。	2023 年 6 月 9 日
1.23.17	eks.8	新的平台版本（包括安全修复和增强功能）。	2023 年 5 月 5 日
1.23.17	eks.7	新的平台版本（包括安全修复和增强功能）。	2023 年 3 月 24 日
1.23.14	eks.6	新的平台版本（包括安全修复和增强功能）。	2023 年 1 月 27 日
1.23.13	eks.5	新的平台版本（包括安全修复和增强功能）。	2022 年 12 月 5 日
1.23.13	eks.4	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 18 日
1.23.12	eks.3	新的平台版本（包括安全修复和增强功能）。	2022 年 11 月 7 日
1.23.10	eks.2	新的平台版本（包括安全修复和增强功能）。	2022 年 9 月 21 日

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

获取当前平台版本

获取集群的当前平台版本（控制台）

1. 打开 Amazon EKS console（Amazon EKS 控制台）。
2. 在导航窗格中，选择集群。
3. 在集群列表中，选择集群名称以检查平台版本。
4. 选择概述选项卡。
5. 可在详细信息部分下方查看平台版本。

获取集群的当前平台版本（AWS CLI）

1. 确定要检查平台版本的集群的名称。
2. 运行以下命令：

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

示例输出如下。

```
"eks.10"
```

Autoscaling

Autoscaling 是一项功能，可以自动扩缩资源以满足您不断变化的需求。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 支持两款自动扩缩产品：

Karpenter

Karpenter 是一款灵活、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。只需不到一分钟时间，Karpenter 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 AWS 相集成，Karpenter 可以即时预置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。有关更多信息，请参阅 Karpenter 文档。

Cluster Autoscaler

当容器组（pod）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 使用自动扩缩组。有关更多信息，请参阅 AWS 上的 Cluster Autoscaler。

管理访问权限

了解如何管理对您的 Amazon EKS 集群的访问权限。使用 Amazon EKS 需要了解 Kubernetes 和 AWS Identity and Access Management（AWS IAM）如何处理访问控制。

本部分包括：

the section called “授予对 Kubernetes API 的访问权限” – 了解如何使应用程序或用户能够对 Kubernetes API 进行身份验证。您可以使用访问条目、aws-auth ConfigMap 或外部 OIDC 提供商。

the section called “使用 kubectl 访问我的集群” – 了解如何将 kubectl 配置为与您的 Amazon EKS 集群进行通信。请使用 AWS CLI 创建 kubeconfig 文件。

the section called “向工作负载授予访问 AWS 的权限” – 了解如何将 Kubernetes 服务账户与 AWS IAM 角色关联。您可以将容器组身份或 IAM 角色用于服务账户的（IRSA）。

常见任务：

- 向开发人员授予对 Kubernetes API 的访问权限。在 AWS Management Console 中查看 Kubernetes 资源。
  - 解决方案：使用访问条目将 Kubernetes RBAC 权限与 AWS IAM 用户或角色相关联。
- 将 kubectl 配置为使用 AWS 凭证与 Amazon EKS 集群通信。
  - 解决方案使用 AWS CLI 创建 kubeconfig 文件。
- 使用外部身份提供商（例如 Ping Identity）对 Kubernetes API 进行用户身份验证。
  - 解决方案：链接外部 OIDC 提供商。
- 授予 Kubernetes 集群上的工作负载调用 AWS API 的能力。
  - 解决方案：使用容器组身份将 AWS IAM 角色与 Kubernetes 服务账户关联。

背景：

- 了解 Kubernetes 服务账户的工作原理。
- 查看 Kubernetes 基于角色的访问控制（RBAC）模型
- 有关管理对 AWS 资源的访问权限的更多信息，请参阅《AWS IAM 用户指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/intro-structure.html。或者，请参加有关使用 AWS IAM 的免费入门培训。

授予对 Kubernetes API 的访问权限

您的集群有一个 Kubernetes API 端点。Kubectl 使用此 API。您可以使用两种类型的身份对此 API 进行身份验证：

- AWS Identity and Access Management（IAM）主体（角色或用户）– 此类型需要对 IAM 进行身份验证。用户可以使用通过身份源提供的凭证以 IAM 用户或联合身份登录到 AWS。如果您的管理员以前使用 IAM 角色设置了身份联合验证，则用户只能使用联合身份登录。当用户使用联合身份验证访问 AWS 时，他们就是在间接分派角色。当用户使用此类身份时，您：
  - 可以为他们分配 Kubernetes 权限，以便其使用集群上的 Kubernetes 对象。有关如何为您的 IAM 主体分配权限以便他们能够访问集群上的 Kubernetes 对象的更多信息，请参阅 管理访问条目。
  - 可以为他们分配 IAM 权限，以便其使用 Amazon EKS API、AWS CLI、AWS CloudFormation、AWS Management Console 或 `eksctl` 使用您的 Amazon EKS 集群及其资源。有关更多信息，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的操作。
  - 节点通过分派 IAM 角色加入您的集群。使用 IAM 主体访问集群的能力由适用于 Kubernetes 的 AWS IAM 身份验证器提供，该工具在 Amazon EKS 控制面板上运行。
- 您自己的 OpenID Connect（OIDC）提供者中的用户 – 此类型需要对您的 OIDC 提供者进行身份验证。有关使用 Amazon EKS 集群设置自己的 OIDC 提供者的更多信息，请参阅 通过 OpenID Connect 身份提供商对集群的用户进行身份验证。当用户使用此类身份时，您：
  - 可以为他们分配 Kubernetes 权限，以便其使用集群上的 Kubernetes 对象。
  - 无法向他们分配 IAM 权限，使他们能够使用 Amazon EKS API、AWS CLI、AWS CloudFormation、AWS Management Console、或 `eksctl` 使用您的 Amazon EKS 集群及其资源。

您可以在集群中使用这两种类型的身份。IAM 身份验证方法无法禁用。OIDC 身份验证方法是可选的。

将 IAM 身份与 Kubernetes 权限相关联

Kubernetes 的 AWS IAM 身份验证器安装在集群的控制面板上。该工具使您允许的 AWS Identity and Access Management（IAM）主体（角色和用户）能够访问集群上的 Kubernetes 资源。您可以使用以下其中一种方法，允许 IAM 主体访问集群上的 Kubernetes 对象：

- 创建访问条目 – 如果您的集群等于或高于集群 Kubernetes 版本的先决条件部分中列出的平台版本，我们建议您使用此选项。

使用访问条目管理集群外的 IAM 主体的 Kubernetes 权限。您可以使用 EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation 和 AWS Management Console 来添加和管理对集群的访问权限。这意味着您可以使用与创建集群相同的工具来管理用户。

要开始使用，请先 设置访问条目，然后 将现有 aws-auth ConfigMap 条目迁移至访问条目。

- 向 **aws-auth ConfigMap** 中添加条目 – 如果您的集群的平台版本早于先决条件部分中列出的版本，则必须使用此选项。如果您的集群的平台版本等于或高于集群 Kubernetes 版本的先决条件部分中列出的平台版本，并且您已向 ConfigMap 中添加了条目，那么我们建议您将这些条目迁移到访问条目。但是，您无法迁移 Amazon EKS 添加到 ConfigMap 的条目，例如与托管节点组一起使用的 IAM 角色的条目或 Fargate 配置文件。有关更多信息，请参阅 the section called “授予对 Kubernetes API 的访问权限”。
  - 如果必须使用 aws-auth ConfigMap 选项，则可以使用 **eksctl create iamidentitymapping** 命令向 ConfigMap 中添加条目。有关更多信息，请参阅 eksctl 文档中的管理 IAM 用户和角色。

设置集群身份验证模式

每个集群都有一种身份验证模式。身份验证模式决定了您可以使用哪些方法来允许 IAM 主体访问集群上的 Kubernetes 对象。有三种身份验证模式。

> **Important**
>
> 一旦启用了访问条目方法，就无法将其禁用。
> 如果在创建集群时未启用 ConfigMap 方法，则以后无法启用它。在引入访问条目之前创建的所有集群都启用了 ConfigMap 方法。

集群内部的 aws-auth ConfigMap

这是 Amazon EKS 集群的原始身份验证模式。创建集群的 IAM 主体是可以使用 kubectl 访问集群的初始用户。初始用户必须将其他用户添加到 aws-auth ConfigMap 中的列表中，并分配影响集群内其他用户的权限。这些其他用户无法管理或移除初始用户，因为 ConfigMap 中没有要管理的条目。

ConfigMap 和访问条目

使用这种身份验证模式，您可以使用这两种方法向集群添加 IAM 主体。请注意，每种方法都存储单独的条目；例如，如果您从 AWS CLI 中添加访问条目，aws-auth ConfigMap 则不会更新。

仅访问条目

使用这种身份验证模式，您可以使用 EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation 和 AWS Management Console 来管理 IAM 主体对集群的访问权限。

每个访问条目都有一种类型，您可以使用访问范围将主体限制在特定的命名空间和访问策略的组合来设置预先配置且可重复使用的权限策略。或者，您可以使用 STANDARD 类型和 Kubernetes RBAC 组来分配自定义权限。

身份验证模式	方法
仅 ConfigMap （CONFIG_MAP ）	aws-auth ConfigMap
EKS API 和 ConfigMap （API_AND_CONFIG_MAP ）	EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation 和 AWS Management Console 中的访问条目以及 aws-auth ConfigMap
仅 EKS API (API)	EKS API、AWS Command Line Interface、AWS SDK、AWS CloudFormation 和 AWS Management Console 中的访问条目

管理访问条目

先决条件

- 熟悉 Amazon EKS 集群的集群访问选项。有关更多信息，请参阅 授予对 Kubernetes API 的访问权限 。
- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。要使用访问条目并更改集群的身份验证模式，集群的平台版本必须与下表中所列的版本相同或更高，或者 Kubernetes 版本必须比下表中所列的版本更高。

Kubernetes 版本	平台版本
1.30	eks.2
1.29	eks.1

Kubernetes 版本	平台版本
1.28	eks.6
1.27	eks.10
1.26	eks.11
1.25	eks.12
1.24	eks.15
1.23	eks.17

您可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令 **aws eks describe-cluster --name *my-cluster* --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'** 来检查当前的 Kubernetes 和平台版本。

> **Important**
>
> 在 Amazon EKS 将您的集群更新到表中列出的平台版本后，Amazon EKS 会为最初创建集群的 IAM 主体创建一个具有集群管理员权限的访问条目。如果您不希望该 IAM 主体拥有集群的管理员权限，请移除 Amazon EKS 创建的访问条目。
>
> 对于平台版本早于上表所列版本的集群，集群创建者始终是集群管理员。无法从创建集群的 IAM 用户或角色中移除集群管理员权限。

- 对您的集群具有以下权限的 IAM 主体：`CreateAccessEntry`、`ListAccessEntries`、`DescribeAccessEntry`、`DeleteAccessEntry` 和 `UpdateAccessEntry`。有关 Amazon EKS 权限的更多信息，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的操作。
- 要为其创建访问条目的现有 IAM 主体，或者要更新或删除的现有访问条目。

设置访问条目

要开始使用访问条目，必须将集群的身份验证模式更改为 `API_AND_CONFIG_MAP` 或 `API` 模式。这为访问条目添加了 API。

AWS Management Console

创建访问条目

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要在其中创建访问条目的集群的名称。
3. 选择访问选项卡。
4. 身份验证模式显示集群的当前身份验证模式。如果模式显示 EKS API，您已经可以添加访问条目了，可以跳过其余的步骤。
5. 选择管理访问。
6. 对于集群身份验证模式，请使用 EKS API 选择一种模式。请注意，您不能将身份验证模式更改回移除 EKS API 和访问条目的模式。
7. 选择 Save changes（保存更改）。Amazon EKS 开始更新集群，集群的状态更改为 Updating，更改记录在更新历史记录选项卡中。
8. 等待集群的状态恢复为 Active。当集群处于 Active 状态时，您可以按照 创建访问条目 中的步骤为 IAM 主体添加对集群的访问权限。

AWS CLI

先决条件

在您的设备上安装并配置了最新版本的 AWS CLI v1 或 AWS CloudShell。AWS CLI v2 已经有一段时间不支持新功能了。您可以使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1` 检查您的当前版本。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI 和使用 `aws configure` 进行快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。要对其进行更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到您的主目录。

1.
2. 运行以下命令。将 *`my-cluster`* 替换为您集群的名称。如果要永久禁用 `ConfigMap` 方法，请将 `API_AND_CONFIG_MAP` 替换为 `API`。

   Amazon EKS 开始更新集群，集群的状态更改为 UPDATING，更改记录在 aws eks list-updates 中。

```
aws eks update-cluster-config --name my-cluster --access-config
 authenticationMode=API_AND_CONFIG_MAP
```

3. 等待集群的状态恢复为 Active。当集群处于 Active 状态时，您可以按照 创建访问条目 中的步骤为 IAM 主体添加对集群的访问权限。

创建访问条目

注意事项

在创建访问条目之前，请考虑以下事项：

- 访问条目包含一个（且仅限一个）现有 IAM 主体的 Amazon 资源名称（ARN）。一个 IAM 主体不能包含在多个访问条目中。您指定的 ARN 的其他注意事项：
  - IAM 最佳实践建议使用具有短期凭证的 IAM 角色而不是具有长期凭证的 IAM 用户访问您的集群。有关更多信息，请参阅《IAM 用户指南》中的 要求人类用户使用带有身份提供者的联合身份验证才能使用临时凭证访问 AWS。
  - 如果 ARN 适用于 IAM 角色，则可以包含路径。`aws-auth ConfigMap` 条目中的 ARN 不能包含路径。例如，您的 ARN 可以是 `arn:aws:iam::111122223333:role/development/apps/my-role` 或 `arn:aws:iam::111122223333:role/my-role`。
  - 如果访问条目的类型不是 STANDARD（请参阅下文关于类型的注意事项），则 ARN 必须位于与您的集群相同的 AWS 账户。如果类型为 STANDARD，则 ARN 可以位于与您的集群所在的账户相同或不同的 AWS 账户。
  - 在创建访问条目后，您将无法更改 IAM 主体。
  - 如果您删除了具有此 ARN 的 IAM 主体，则访问条目不会自动删除。对于已删除的 IAM 主体，我们建议您删除具有 ARN 的访问条目。如果您不删除访问条目并重新创建 IAM 主体，即使该访问条目具有相同的 ARN，也将无法运行。这是因为尽管重新创建的 IAM 主体的 ARN 相同，但对于重新创建的 IAM 主体，`roleID` 或 `userID`（您可以用 `aws sts get-caller-identity` AWS CLI 命令查看）与原始 IAM 主体不同。即使您看不到 IAM 主体的 `roleID` 或 `userID` 访问条目，Amazon EKS 也会将其与访问条目一起存储。
- 每个访问条目都有一种类型。您可以将 `EC2 Linux`（用于与 Linux 或 Bottlerocket 自行管理的节点一起使用的 IAM 角色）、`EC2 Windows`（用于与 Windows 自行管理的节点一起使用的 IAM 角色）、`FARGATE_LINUX`（用于与 AWS Fargate (Fargate) 一起使用的 IAM 角色）或 `STANDARD` 指定为一种类型。如果您不指定类型，则 Amazon EKS 会自动将类型设置为 `STANDARD`。无需为用于托管节点组或 Fargate 配置文件的 IAM 角色创建访问条目，因为无论您的集群处于哪个平台版本，Amazon EKS 都会将这些角色的条目添加到 `aws-auth ConfigMap` 中。

在创建访问条目后，您将无法更改类型。

- 如果访问条目的类型为 STANDARD，则可以为该访问条目指定用户名。如果您没有为用户名指定值，Amazon EKS 会根据访问条目的类型以及您指定的 IAM 主体是 IAM 角色还是 IAM 用户，为您设置以下值之一。除非您出于特定原因要指定自己的用户名，否则我们建议您不要指定用户名，而是让 Amazon EKS 为您自动生成用户名。如果您指定自己的用户名：
  - 不能以 `system:`、`eks:`、`aws:`、`amazon:` 或 `iam:` 开头。
  - 如果该用户名用于 IAM 角色，我们建议您在用户名的末尾添加 `{{SessionName}}`。如果您在用户名中添加 `{{SessionName}}`，则该用户名必须在 {{SessionName}} 之前加一个冒号。如果分派此角色，则分派此角色时指定的会话名称将自动传递到集群，并显示在 CloudTrail 日志中。例如，您不能将用户名设置为 `john{{SessionName}}`。用户名必须是 `:john{{SessionName}}` 或 `jo:hn{{SessionName}}`。冒号只能在 `{{SessionName}}` 前面。下表中由 Amazon EKS 生成的用户名包含一个 ARN。由于 ARN 包含冒号，因此符合此要求。如果用户名中未包含 `{{SessionName}}`，则不需要使用冒号。

IAM 主体类型	类型	Amazon EKS 自动设置的用户名值
用户	STANDARD	用户的 ARN。例如：`arn:aws:iam::` *`111122223333`* `:user/`*`my-user`*
角色	STANDARD	分派角色时此角色的 STS ARN。Amazon EKS 会将 `{{SessionName}}` 附加到该角色中。 例如：`arn:aws:sts::` *`111122223333`* `:assumed-role/`*`my-role`*`/{{SessionName}}` 如果您指定的角色的 ARN 包含路径，Amazon EKS 会在生成的用户名中将其移除。

IAM 主体类型	类型	Amazon EKS 自动设置的用户名值
角色	EC2 Linux 或 EC2 Windows	system:node:{{EC2PrivateDNSName}}
角色	FARGATE_LINUX	system:node:{{SessionName}}

创建访问条目后，您可以更改用户名。

- 如果访问条目的类型为 STANDARD，并且您想使用 Kubernetes RBAC 授权，则可以在访问条目中添加一个或多个组名。创建访问条目后，您可以添加和移除组名。要让 IAM 主体能够访问集群上的 Kubernetes 对象，您必须创建和管理 Kubernetes 基于角色的授权（RBAC）对象。在集群上创建 Kubernetes RoleBinding 或 ClusterRoleBinding 对象，将组名指定为 kind: Group 的 subject。Kubernetes 授权 IAM 主体访问您在 Kubernetes Role 或 ClusterRole 对象中指定的任何集群对象，这些对象也在绑定的 roleRef 中指定。如果您指定组名，我们建议您熟悉 Kubernetes 基于角色的授权（RBAC）对象。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

  > **Important**
  >
  > Amazon EKS 不会确认您的集群上存在的任何 Kubernetes RBAC 对象是否包含您指定的任何组名。

  或者除了 Kubernetes 授权 IAM 主体访问集群上的 Kubernetes 对象之外，您可以将 Amazon EKS 访问策略关联到访问条目。Amazon EKS 授权 IAM 主体使用访问策略中的权限访问您集群上的 Kubernetes 对象。您可以将访问策略的权限范围限定到您指定的 Kubernetes 命名空间。使用访问策略不需要您管理 Kubernetes RBAC 对象。有关更多信息，请参阅 将访问策略与访问条目关联和取消关联。
- 如果您创建类型为 EC2 Linux 或 EC2 Windows 的访问条目，则创建访问条目的 IAM 主体必须拥有 iam:PassRole 权限。有关更多信息，请参阅《IAM 用户指南》中的向用户授予权限以将角色传递给 AWS 服务。
- 与标准 IAM 行为类似，访问条目创建和更新最终是一致的，并且在初始 API 调用成功返回后可能需要几秒钟才能生效。您在设计应用程序时，必须考虑到这些可能的延迟。在应用程序的关键、高可用

性代码路径中，我们不建议创建或更新访问条目。而应在不常运行的、单独的初始化或设置例程中进行 更改。另外，在生产工作流程依赖这些更改之前，请务必验证更改已传播。

- 访问条目不支持服务相关角色。如果主体 ARN 是服务相关角色，则无法创建访问条目。您可以通过服务相关角色的 ARN（格式为 `arn:aws:iam::*:role/aws-service-role/*`）来识别它们。

您可以使用 AWS Management Console 或 AWS CLI 创建访问条目。

AWS Management Console

创建访问条目

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要在其中创建访问条目的集群的名称。
3. 选择访问选项卡。
4. 选择创建访问条目。
5. 对于 IAM 主体，请选择现有的 IAM 角色或用户。IAM 最佳实践建议使用具有短期凭证的 IAM 角色而不是具有长期凭证的 IAM 用户访问您的集群。有关更多信息，请参阅《IAM 用户指南》中的 要求人类用户使用带有身份提供者的联合身份验证才能使用临时凭证访问 AWS。
6. 对于类型，如果访问条目是用于自行管理的 Amazon EC2 节点的节点角色，请选择 EC2 Linux 或 EC2 Windows。否则，请接受默认值（标准）。
7. 如果您选择的类型是标准，并且想要指定用户名，请输入用户名。
8. 如果您选择的类型是标准，并且您想对 IAM 主体使用 Kubernetes RBAC 授权，请为组指定一个或多个名称。如果您未指定任何组名并想使用 Amazon EKS 授权，则可以在后续步骤中或创建访问条目之后关联访问策略。
9. （可选）对于标签，为访问条目分配标签。例如，为了更轻松地查找所有具有相同标签的资源。
10. 选择下一步。
11. 在添加访问策略页面上，如果您选择的类型是标准，并且希望 Amazon EKS 授权 IAM 主体拥有对集群上 Kubernetes 对象的权限，请完成以下步骤。否则，请选择下一步。

    a. 对于策略名称，选择访问策略。您无法查看访问策略的权限，但其包含的权限与 Kubernetes 面向用户的 `ClusterRole` 对象中的权限类似。有关更多信息，请参阅 Kubernetes 文档中的面向用户的角色。

    b. 请选择以下选项之一：

- 集群 – 如果您希望 Amazon EKS 授权 IAM 主体拥有集群上所有 Kubernetes 对象的访问策略权限，请选择此选项。
- Kubernetes 命名空间 – 如果您希望 Amazon EKS 授权 IAM 主体拥有集群上特定 Kubernetes 命名空间内所有 Kubernetes 对象的访问策略权限，请选择此选项。对于命名空间，输入集群上 Kubernetes 命名空间的名称。如果要添加其他命名空间，请选择添加新命名空间并输入命名空间名称。

c. 如果要添加其他策略，请选择添加策略。您可以对每个策略设定不同的范围，但每个策略只能添加一次。

d. 选择下一步。

12. 查看访问条目的配置。如果有任何内容看起来不正确，请选择上一步以返回步骤并更正错误。如果配置正确，请选择创建。

AWS CLI

先决条件

在您的设备上安装并配置了最新版本的 AWS CLI v1 或 AWS CloudShell。AWS CLI v2 已经有一段时间不支持新功能了。您可以使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1` 检查您的当前版本。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI 和使用 `aws configure` 进行快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。要对其进行更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到您的主目录。

创建访问条目

您可以使用以下任意示例来创建访问条目：

- 为自行管理的 Amazon EC2 Linux 节点组创建访问条目。将 *my-cluster* 替换为您的集群名称，将 *111122223333* 替换为您的 AWS 账户 ID，将 *EKS-my-cluster-self-managed-ng-1* 替换为您的节点 IAM 角色的名称。如果您的节点组是 Windows 节点组，则将 *EC2_Linux* 替换为 **EC2_Windows**。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_Linux
```

当您指定除 STANDARD 之外的类型时，不能使用 --kubernetes-groups 选项。您无法将访问策略与该访问条目相关联，因为其类型的值不是 STANDARD。

- 创建一个允许 IAM 角色的访问条目，该角色不用于 Amazon EC2 自行管理的节点组，您想要 Kubernetes 通过该角色授权访问您的集群。将 *my-cluster* 替换为您的集群的名称，将 *111122223333* 替换为您的 AWS 账户 ID，将 *my-role* 替换为您的 IAM 角色的名称。将 *Viewers* 替换为您在集群上的 Kubernetes RoleBinding 或 ClusterRoleBinding 对象中指定的组的名称。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --
kubernetes-groups Viewers
```

- 创建允许 IAM 用户对集群进行身份验证的访问条目。之所以提供此示例，是因为尽管 IAM 最佳实践建议使用具有短期凭证的 IAM 角色而不是具有长期凭证的 IAM 用户访问集群，但这仍是可行的。有关更多信息，请参阅《IAM 用户指南》中的 要求人类用户使用带有身份提供者的联合身份验证才能使用临时凭证访问 AWS。

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

如果您希望此用户对您的集群拥有的访问权限超过 Kubernetes API 发现角色中的权限，则需要将访问策略与访问条目相关联，因为未使用 --kubernetes-groups 选项。有关更多信息，请参阅 Kubernetes 文档中的 将访问策略与访问条目关联和取消关联 和 API 发现角色。

更新访问条目

您可以使用 AWS Management Console 或 AWS CLI 更新访问条目。

AWS Management Console

更新访问条目

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要在其中创建访问条目的集群的名称。
3. 选择访问选项卡。
4. 选择要更新的访问条目。

5. 选择编辑。
6. 对于用户名，您可以更改现有值。
7. 对于组，您可以移除现有的组名称或添加新的组名称。如果存在以下组名称，请不要将其移除：system:nodes 或 system:bootstrappers。移除这些组可能会导致集群无法正常运行。如果您未指定任何组名称并想使用 Amazon EKS 授权，请在后续步骤中关联访问策略。
8. 对于标签，您可以为访问条目分配标签。例如，为了更轻松地查找所有具有相同标签的资源。您也可以移除现有标签。
9. 选择 Save changes（保存更改）。
10. 如果要将访问策略与条目关联，请参阅 将访问策略与访问条目关联和取消关联。

AWS CLI

先决条件

在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

更新访问条目

将 *my-cluster* 替换为您的集群名称，将 *111122223333* 替换为您的 AWS 账户 ID，将 *EKS-my-cluster-my-namespace-Viewers* 替换为 IAM 角色的名称。

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

如果访问条目的类型值不是 STANDARD，则不能使用 --kubernetes-groups 选项。您也不能将访问策略与访问条目关联到 STANDARD 类型之外的访问条目。

删除访问条目

如果您发现自己误删了访问条目，则可以随时重新创建。如果您要删除的访问条目与任何访问策略相关联，则关联会自动删除。在删除访问条目之前，您不必取消访问策略与访问条目的关联。

您可以使用 AWS Management Console 或 AWS CLI 删除访问条目。

AWS Management Console

删除访问条目

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择要从中删除访问条目的集群的名称。
3. 选择访问选项卡。
4. 在访问条目列表中，选择要删除的访问条目。
5. 选择 Delete。
6. 在确认对话框中，选择删除。

AWS CLI

先决条件

在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

删除访问条目

将 *`my-cluster`* 替换为您的集群名称，将 *`111122223333`* 替换为您的 AWS 账户 ID，将 *`my-role`* 替换为您不想再访问集群的 IAM 角色的名称。

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

将访问策略与访问条目关联和取消关联

您可以为类型为 `STANDARD` 的访问条目分配一个或多个访问策略。Amazon EKS 会自动向其他类型的访问条目授予在您的集群中正常运行所需的权限。Amazon EKS 访问策略包含 Kubernetes 权限，但不

包含 IAM 权限。在将访问策略关联到访问条目之前，请确保您熟悉每个访问策略中包含的 Kubernetes 权限。有关更多信息，请参阅 访问策略权限。如果所有访问策略都不符合您的要求，则不要将访问策略与访问条目相关联。而是为访问条目指定一个或多个组名称，然后创建和管理 Kubernetes 基于角色的访问控制对象。有关更多信息，请参阅 创建访问条目。

先决条件

- 现有的访问条目。要创建该文件，请参阅 创建访问条目。
- 具有以下权限的 AWS Identity and Access Management 角色或用户：`ListAccessEntries`、`DescribeAccessEntry`、`UpdateAccessEntry`、`ListAccessPolicie` 和 `DisassociateAccessPolicy`。有关更多信息，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的操作。

在将访问策略与访问条目关联之前，请考虑以下要求：

- 您可以将多个访问策略关联到每个访问条目，但只能将每个策略与一个访问条目关联一次。如果您关联多个访问策略，则访问条目的 IAM 主体拥有所有关联访问策略中包含的所有权限。
- 您可以将访问策略的范围限定为集群上的所有资源，也可以指定一个或多个 Kubernetes 命名空间的名称。您可以为命名空间名称使用通配符。例如，如果要将访问策略的范围限定为以 `dev-` 开头的所有命名空间，则可以指定 `dev-*` 为命名空间名称。请确保您的集群上存在命名空间，并且拼写与集群上的实际命名空间名称相匹配。Amazon EKS 不会确认您的集群上命名空间的拼写或是否存在。
- 将访问策略与访问条目关联后，您可以更改访问策略的访问范围。如果您已将访问策略的范围限定为 Kubernetes 命名空间，则可以根据需要为关联添加和移除命名空间。
- 如果您将访问策略关联到同时指定了组名称的访问条目，则 IAM 主体拥有所有关联访问策略中的所有权限。该主体还拥有在指定组名称的任何 Kubernetes `Role` 和 `RoleBinding` 对象中指定的任何 Kubernetes `Role` 或 `ClusterRole` 对象中的所有权限。
- 如果您运行 `kubectl auth can-i --list` 命令，则不会看到访问策略分配的任何 Kubernetes 权限，这些权限与您在运行命令时使用的 IAM 主体的访问条目相关联。只有在您已经在 Kubernetes `Role` 或 `ClusterRole` 对象中授予了 Kubernetes 权限，并且已经将这些权限绑定到为访问条目指定的组名称或用户名时，该命令才会显示这些权限。
- 如果您在与集群上的 Kubernetes 对象交互时伪装成 Kubernetes 用户或组，例如使用带 **`--as`** ***`username`*** 或 **`--as-group`** ***`group-name`*** 的 `kubectl` 命令，则表示您强制使用 Kubernetes RBAC 授权。因此，IAM 主体没有与访问条目关联的任何访问策略所分配的权限。IAM 主体伪装的用户或组拥有的唯一 Kubernetes 权限是您在 Kubernetes `Role` 或 `ClusterRole` 对象中授予他们的 Kubernetes 权限，这些权限已绑定到组名称或用户名。要使您的 IAM 主体拥有相关访问策略中的权限，请不要伪装成 Kubernetes 用户或组。IAM 主体仍将拥有您在 Kubernetes `Role` 或

`ClusterRole` 对象中授予他们的任何权限，这些权限已绑定到您为访问条目指定的组名称或用户名。有关更多信息，请参阅 Kubernetes 文档中的用户伪装。

您可以使用 AWS Management Console 或 AWS CLI 将访问策略与访问条目相关联。

AWS Management Console

使用 AWS Management Console 将访问策略与访问条目关联

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 选择具有要与访问策略关联的访问条目的集群的名称。
3. 选择访问选项卡。
4. 如果访问条目的类型为标准，则可以关联或取消关联 Amazon EKS 访问策略。如果您的访问条目类型不是标准，则此选项不可用。
5. 选择关联访问策略。
6. 对于策略名称，选择具有您希望 IAM 主体拥有的权限的策略。要查看每个策略包含的权限，请参阅 访问策略权限。
7. 对于访问范围，选择一个访问范围。如果选择集群，则访问策略中的权限将授予 IAM 主体访问所有 Kubernetes 命名空间中的资源。如果选择 Kubernetes 命名空间，则可以选择添加新命名空间。在出现的命名空间字段中，您可以输入集群上 Kubernetes 命名空间的名称。如果您希望 IAM 主体拥有跨多个命名空间的权限，则可以输入多个命名空间。
8. 选择添加访问策略。

AWS CLI

先决条件

在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

将访问策略与访问条目关联

1. 查看可用的访问策略。

```
aws eks list-access-policies --output table
```

示例输出如下。

```
-------------------------------------------------------------------------------------
|                                        ListAccessPolicies
                       |
+-------------------------------------------------------------------------------------
+
||                                          accessPolicies
                      ||
|+-------------------------------------------------------------------
+-------------------------------+|
||                                arn                                 |
     name              ||
|+-------------------------------------------------------------------
+-------------------------------+|
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy        |
 AmazonEKSAdminPolicy         ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-------------------------------------------------------------------
+-------------------------------+|
```

要查看每个策略包含的权限，请参阅 访问策略权限。

2. 查看您现有的访问条目。将 *my-cluster* 替换为您集群的名称。

```
aws eks list-access-entries --cluster-name my-cluster
```

示例输出如下。

```
{
    "accessEntries": [
```

```
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
}
```

3. 将访问策略与访问条目关联。以下示例将 AmazonEKSViewPolicy 访问策略与访问条目关联。每当 *my-role* IAM 角色尝试访问集群上的 Kubernetes 对象时，Amazon EKS 都将授权该角色使用策略中的权限仅访问 *my-namespace1* 和 *my-namespace2* Kubernetes 命名空间中的 Kubernetes 对象。将 *my-cluster* 替换为您的集群名称，将 *111122223333* 替换为您的 AWS 账户 ID，将 *my-role* 替换为您希望 Amazon EKS 授权其访问 Kubernetes 集群对象的 IAM 角色的名称。

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --
policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

如果您希望 IAM 主体拥有整个集群的权限，请将 **type=namespace,namespaces=*my-namespace1*,*my-namespace2*** 替换为 **type=cluster**。如果要将多个访问策略关联到访问条目，请多次运行该命令，每个访问条目都有唯一的访问策略。每个关联的访问策略都有自己的范围。

> **Note**
>
> 如果以后要更改关联访问策略的范围，请使用新的范围再次运行前面的命令。例如，如果您想移除 *my-namespace2*，则只能使用 **type=namespace,namespaces=*my-namespace1*** 再次运行该命令。如果要将范围从 **namespace** 更改为 **cluster**，则可以使用 **type=cluster** 再次运行该命令，移除 **type=namespace,namespaces=*my-namespace1*,*my-namespace2***。

取消访问策略与访问条目的关联

1. 确定哪些访问策略与访问条目关联。

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-
arn arn:aws:iam::111122223333:role/my-role
```

示例输出如下。

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws:iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

在前面的示例中，此访问条目的 IAM 主体拥有集群上所有命名空间的查看权限，以及对两个 Kubernetes 命名空间的管理员权限。

2. 取消访问策略与访问条目的关联。在此示例中，AmazonEKSAdminPolicy 策略与访问条目取消关联。但是，IAM 主体保留 AmazonEKSViewPolicy 访问策略中对 *my-namespace1* 和 *my-namespace2* 命名空间中对象的权限，因为该访问策略并未与访问条目取消关联。

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

访问策略权限

访问策略包括包含 Kubernetes `verbs`（权限）和 `resources` 的 `rules`。访问策略不包括 IAM 权限或资源。与 Kubernetes `Role` 和 `ClusterRole` 对象类似，访问策略仅包括 `allow` `rules`。您无法修改访问策略的内容。您无法创建自己的访问策略。如果访问策略中的权限不符合您的需求，请创建 Kubernetes RBAC 对象并为访问条目指定组名称。有关更多信息，请参阅 创建访问条目。访问策略中包含的权限与 Kubernetes 面向用户的集群角色中的权限类似。有关更多信息，请参阅 Kubernetes 文档中的面向用户的角色。

选择任意访问策略以查看其内容。每个访问策略中每个表的每一行都是一个单独的规则。

AmazonEKSAdminPolicy

此访问策略包括授予 IAM 主体对资源的大部分权限的权限。当与访问条目关联时，其访问范围通常是一个或多个 Kubernetes 命名空间。如果您希望 IAM 主体拥有集群上所有资源的管理员访问权限，请改为将 AmazonEKSClusterAdminPolicy 访问策略关联到您的访问条目。

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
apps	daemonsets, deployments, deployments/rollback, deployments/scale, replicasets, replicasets/scale, statefulsets, statefulsets/scale	create, delete, deletecollection, patch, update
apps	controllerrevisions, daemonsets, daemonsets/status, deployments, deployments/scale, deployments/status, replicasets, replicasets/scale, replicasets/status, statefulsets, statefulsets/	get, list, watch

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
	scale , statefulsets/status	
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get,list, watch

AmazonEKSClusterAdminPolicy

此访问策略包括授予 IAM 主体对集群的管理员访问权限的权限。当与访问条目关联时，其访问范围通常是集群，而不是 Kubernetes 命名空间。如果您希望 IAM 主体的管理范围更有限，请考虑将 AmazonEKSAdminPolicy 访问策略与您的访问条目相关联。

ARN – arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy

Kubernetes API 组	Kubernetes nonResourceURL	Kubernetes 资源	Kubernetes 动词（权限）
*		*	*
	*		*

AmazonEKSAdminViewPolicy

此访问策略包括授予 IAM 主体列出/查看集群中所有资源的权限。请注意，这包括 Kubernetes Secrets。

ARN – arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
*	*	get, list, watch

AmazonEKSEditPolicy

此访问策略包括允许 IAM 主体编辑大多数 Kubernetes 资源的权限。

ARN – arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
networking.k8s.io	ingresses , ingresses /status , networkpo licies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecol lection , patch, update
policy	poddisruptionbudgets , poddisruptionbudge ts/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/port forward , pods/proxy	create, delete, deletecol lection , patch, update
	configmaps , events, persistentvolumecl aims , replicati oncontrollers , replicationcontrol lers/scale , secrets, serviceaccounts , services, services/ proxy	create, delete, deletecol lection , patch, update

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

AmazonEKSViewPolicy

此访问策略包括允许 IAM 主体查看大多数 Kubernetes 资源的权限。

ARN – arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicase	get, list, watch

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
	ts/status ，statefuls ets ，statefulsets/ scale ，statefulsets/ status	
autoscaling	horizontalpodautos calers ，horizonta lpodautoscalers/st atus	get, list, watch
batch	cronjobs, cronjobs/ status ，jobs, jobs/stat us	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets 、daemonset s/status 、deploymen ts 、deployments/ scale、deployments/ status 、ingresses 、ingresses/ status、networkpo licies 、replicase ts 、replicasets/ scale、replicasets/ status 、replicati oncontrollers/scale	get, list, watch
networking.k8s.io	ingresses ，ingresses /status ，networkpo licies	get, list, watch
policy	poddisruptionbudgets ， poddisruptionbudge ts/status	get, list, watch

Kubernetes API 组	Kubernetes 资源	Kubernetes 动词（权限）
	configmaps ，endpoints ，persistentvolumeclaims ，persistentvolumeclaims/status ，pods, replicationcontrollers ，replicationcontrollers/scale ，serviceaccounts ，services, services/status	get, list, watch
	bindings、events、limitranges 、namespaces/status、pods/log、pods/status 、replicationcontrollers/status 、resourcequotas 、resourcequotas/status	get, list, watch
	namespaces	get, list, watch

访问策略更新

查看有关自引入访问策略以来更新的详细信息。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	描述	日期
添加 AmazonEKSAdminViewPolicy	添加新策略以扩展查看权限，包括 Secrets 等资源。	2024 年 4 月 23 日
引入访问策略。	Amazon EKS 引入访问策略。	2023 年 5 月 29 日

将现有 aws-auth ConfigMap 条目迁移至访问条目

如果您已向集群的 aws-auth ConfigMap 中添加了条目，我们建议您为 aws-auth ConfigMap 中的现有条目创建访问条目。创建访问条目后，您可以从 ConfigMap 中移除这些条目。您无法将访问策略与 aws-auth ConfigMap 中的条目关联起来。如果您想将访问策略与您的 IAM 主体相关联，请创建访问条目。

> **Important**
>
> 不要移除在您向集群添加托管节点组或 Fargate 配置文件时由 Amazon EKS 创建的现有 aws-auth ConfigMap 条目。如果您移除 Amazon EKS 在 ConfigMap 中创建的条目，您的集群将无法正常运行。但是，您可以在为自行管理的节点组创建访问条目后移除这些节点组对应的任何条目。

先决条件

- 熟悉访问条目和访问策略。有关更多信息，请参阅管理访问条目 和将访问策略与访问条目关联和取消关联。
- 一个现有集群，其平台版本等于或高于允许 IAM 角色或用户访问您的 Amazon EKS 集群上的 Kubernetes 对象主题的“先决条件”中列出的版本。
- 您的设备或 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。
- 用于修改 kube-system 命名空间中的 aws-auth ConfigMap 的 Kubernetes 权限。
- 具有以下权限的 AWS Identity and Access Management 角色或用户：CreateAccessEntry 和 ListAccessEntries。有关更多信息，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的操作。

将条目从您的 **aws-auth ConfigMap** 迁移到访问条目

1. 查看您的 aws-auth ConfigMap 中的现有条目。将 *my-cluster* 替换为您集群的名称。

```
eksctl get iamidentitymapping --cluster my-cluster
```

示例输出如下。

```
ARN
              USERNAME                                      GROUPS
                            ACCOUNT
arn:aws:iam::111122223333:role/EKS-my-cluster-Admins
              Admins                                        system:masters
arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
              my-namespace-Viewers                          Viewers
arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
                              system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:user/my-user
              my-user
arn:aws:iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws:iam::111122223333:role/EKS-my-cluster-managed-ng
                              system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. 为您创建的并在上一个输出中返回的任何 ConfigMap 条目创建访问条目。创建访问条目时，请确保为输出中返回的 ARN、USERNAME、GROUPS 和 ACCOUNT 指定相同的值。在此示例输出中，您将为除最后两个条目外的其他所有条目创建访问条目，因为最后两个条目是由 Amazon EKS 为 Fargate 配置文件和托管节点组创建的。
3. 从 ConfigMap 中删除您创建的任何访问条目对应的条目。如果您不从 ConfigMap 中删除条目，则 IAM 主体 ARN 的访问条目设置将覆盖 ConfigMap 条目。将 *111122223333* 替换为您的 AWS 账户 ID，将 *EKS-my-cluster-my-namespace-Viewers* 替换为您的 ConfigMap 中相应条目的角色名称。如果您要移除的条目是用于 IAM 用户而不是 IAM 角色的，请将 **role** 替换为 **user**，将 *EKS-my-cluster-my-namespace-Viewers* 替换为用户名。

```
eksctl delete iamidentitymapping --arn arn:aws:iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

让 IAM 主体访问您的集群

> **Important**
>
> aws-auth ConfigMap 已弃用。管理对 Kubernetes API 的访问权限的建议方法是访问条目。

Amazon EKS 控制面板上运行的 AWS IAM Authenticator for Kubernetes 支持使用 IAM 主体访问集群。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 主体添加到 Amazon EKS 集群

创建 Amazon EKS 集群时，将为创建集群的 IAM 主体自动授予 Amazon EKS 控制面板中基于集群角色的访问控制（RBAC）配置中的 system:masters 权限。该主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主体。要授予其他 IAM 主体与集群进行交互的能力，请编辑 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 Kubernetes 基于角色的访问控制（RBAC）配置的更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

要将 IAM 主体添加到 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
   ```

   示例输出如下。

   ```
   [...]
   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 IAM 主体有权访问集群。IAM 最佳实践建议您向角色而不是用户授予权限。您可以使用以下命令查看哪些其他主体当前有权访问您的集群：

```
kubectl describe -n kube-system configmap/aws-auth
```

示例输出如下。

```
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 或者 clusterroles 和 clusterrolebindings，您可以将 IAM 主体映射到其中。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 主体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
   ```

   示例输出如下。

   ```
   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
```

示例输出如下。

```
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 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。

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 AWS 区域。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

示例输出如下。

```
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 对象中指定的组名称。请将 *111122223333* 替换为您的账户 ID。您可以将 *admin* 替换为您选择的任何名称。

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-role --username admin --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。

示例输出如下。

```
[...]
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。IAM 最佳实践建议您向角色而不是用户授予权限。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 对象中指定的组名称。请将 *111122223333* 替换为您的账户 ID。您可以将 *my-user* 替换为您选择的任何名称。

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user --username my-user --
group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

示例输出如下。

```
[...]
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
```

示例输出如下。

```
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/admin
                        my-role                                 eks-console-
dashboard-full-access-group
arn:aws:iam::111122223333:user/my-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 中的过程应用库存 ConfigMap。

2. 将您的 IAM 主体添加到 ConfigMap。IAM 组不是 IAM 主体，因此无法将其添加到 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。
   - 要添加 IAM 用户：IAM 最佳实践建议您向角色而不是用户授予权限。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 所需的权限。

- 一个 mapUsers 部分，此部分将默认 AWS 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 AWS 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 所需的权限。

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-authConfigMap 并应用于集群。最初创建它的目的是允许节点加入您的集群，也可以使用 ConfigMap 为 IAM 主体添加基于角色的访问控

制（RBAC）。如果您尚未启动自行管理的节点并且未将 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了 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 AWS 身份验证器配置映射。

   a. 下载配置映射。

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-
      eks/cloudformation/2020-10-29/aws-auth-cm.yaml
      ```

   b. 在 aws-auth-cm.yaml 文件中，将 rolearn 设置为与您的节点关联的 IAM 角色的 Amazon 资源名称（ARN）。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
      ```

      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。

      您可以检查节点组的 AWS 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 AWS Management Console 中使用 Amazon EKS 提供的 AWS 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，请参阅故障排除主题中的 未经授权或访问被拒绝 (kubectl)。

3. 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。

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）身份提供者作为对您的集群的用户进行身份验证的方法。OIDC 身份提供者可与 AWS 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the section called “授予对 Kubernetes API 的访问权限”。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 身份提供者与您的集群关联。
- Kubernetes 没有提供 OIDC 身份提供者。您可以使用现有的公共 OIDC 身份提供者，也可以运行您自己的身份提供者。有关经认证提供商的列表，请参阅 OpenID 网站上的 OpenID 认证。
- OIDC 身份提供者的发布者 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 身份提供者。
- 您不能在集群上禁用 IAM 身份验证，因为在将节点加入集群时仍需要使用。
- Amazon EKS 集群仍必须由 AWS IAM 主体创建，而不是由 OIDC 身份提供者用户创建。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。
- 如果为控制面板启用了 CloudWatch 日志，则将在集群的审核日志中列出经 OIDC 身份提供者验证的用户。有关更多信息，请参阅 启用和禁用控制层面日志。

- 您无法使用来自 OIDC 提供者的账户登录 AWS Management Console。您只能通过使用 AWS Identity and Access Management 账户登录 AWS Management Console，从而在控制台中查看 Kubernetes 资源。

关联 OIDC 身份提供者

您需要提供者提供以下信息，然后才能将 OIDC 身份提供者与集群关联：

发布者 URL

OIDC 身份提供者的 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 AWS Management Console 关联身份提供商。

eksctl

使用 **`eksctl`** 将 OIDC 身份提供者与您的集群关联

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 身份提供者处获取的。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 身份提供者搭配使用，请参阅 Kubernetes 文档中的使用 kubectl。

AWS Management Console

使用 AWS Management Console 将 OIDC 身份提供者与您的集群关联

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，然后选择访问选项卡。
3. 在 OIDC 身份提供者部分中，选择关联身份提供者。
4. 在关联 OIDC 身份提供者页面上，输入或选择以下选项，然后选择关联。
   - 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。
   - 对于客户端 ID，输入 OIDC 身份提供者的客户端 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。

5. 要使用 `kubectl` 与您的集群和 OIDC 身份提供者搭配使用，请参阅 Kubernetes 文档中的使用 `kubectl`。

取消 OIDC 身份提供者与集群的关联

如果您取消 OIDC 身份提供者与集群的关联，则提供者中包含的用户将无法再访问该集群。但是，您仍然可以使用 IAM 主体访问该集群。

使用 AWS Management Console 取消 OIDC 身份提供者与集群的关联

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在 OIDC 身份提供者部分中，选择取消关联，输入身份提供者名称，然后选择 `Disassociate`。

示例 IAM 策略

如果要阻止 OIDC 身份提供者与集群关联，请创建以下 IAM 策略并将其与 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 策略和添加 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:us-west-2.amazonaws.com:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

如果 clientID 为 kubernetes 而 issuerUrl 为 https://cognito-idp.us-west-2amazonaws.com/*，则以下示例策略允许进行 OIDC 身份提供者关联。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
```

```
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

经合作伙伴验证的 OIDC 身份提供商

Amazon EKS 与为兼容的 OIDC 身份提供商提供支持的合作伙伴网络保持着合作关系。参考以下合作伙伴的文档，了解如何将身份提供商与 Amazon EKS 集成的详细信息。

合作伙伴	产品	文档
PingIdentity	企业版 PingOne	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。如果您开发的商业支持的 OIDC 兼容身份提供商未在此处列出，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。

为 Amazon EKS 集群创建或更新 `kubeconfig` 文件

在本主题中，您将为您的集群创建 `kubeconfig` 文件（或更新现有文件）。

`kubectl` 命令行工具使用 `kubeconfig` 文件中的配置信息与集群的 API 服务器通信。有关更多信息，请参阅 Kubernetes 文档中的 Organizing Cluster Access Using kubeconfig Files（使用 kubeconfig 文件组织集群访问权限）。

Amazon EKS 使用带 `kubectl` 的 `aws eks get-token` 命令进行集群身份验证。默认情况下，AWS CLI 使用以下命令返回的相同凭证：

```
aws sts get-caller-identity
```

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 有权将 `eks:DescribeCluster` API 操作用于您指定的集群的 IAM 用户或角色。有关更多信息，请参阅 Amazon EKS 基于身份的策略示例。如果您使用自己的 OpenID Connect 提供者提供的身份来访问您的集群，请参阅 Kubernetes 文档中的使用 `kubectl` 来创建或更新您的 `kube config` 文件。

自动创建 **kubeconfig** 文件

先决条件

- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 将 `eks:DescribeCluster` API 操作用于您指定的集群的权限。有关更多信息，请参阅 Amazon EKS 基于身份的策略示例。

使用 AWS CLI 创建 **kubeconfig**

1. 为集群创建或更新 kubeconfig 文件。将 *region-code* 替换为您的集群所在的 AWS 区域，并将 *my-cluster* 替换为您的集群的名称。

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

   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 AWS CLI 或开发工具包凭证链中的 IAM 主体。通过运行 `aws sts get-caller-identity` 命令可以查看默认 AWS CLI 或开发工具包标识。

   对于所有可用选项，运行 `aws eks update-kubeconfig help` 命令，或请参阅《AWS CLI Command Reference》中的 update-kubeconfig。

2. 测试配置。

   ```
   kubectl get svc
   ```

   示例输出如下。

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

   如果您收到任何授权或资源类型错误，请参阅故障排除主题中的 未经授权或访问被拒绝 (kubectl)。

使用 Kubernetes 服务账户授予 Kubernetes 工作负载访问 AWS 的权限

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 AWS 服务，您可以将服务账户映射到 AWS 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。

服务账户令牌

Kubernetes 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早期客户端，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 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 服务器会拒绝令牌超过 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 服务器收到令牌超过一小时的请求时，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，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（7,776,000 秒）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。我们建议确保已在集群上安装列出的版本或更高版本。

- Amazon VPC CNI plugin for Kubernetes 和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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件 和 cni-metrics-helper。
- 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 使用 CoreDNS Amazon EKS 附加组件。
- AWS 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 AWS Load Balancer Controller 是什么？。
- 当前 `kube-proxy` 版本。要查看当前版本或更新它，请参阅 使用 Kubernetes `kube-proxy` 附加组件。
- AWS 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。

向 Amazon Elastic Kubernetes Service 集群上的工作负载授予 AWS Identity and Access Management 权限

对于向 Amazon EKS 集群中运行的工作负载授予 AWS Identity and Access Management 权限，Amazon EKS 提供了两种方法：服务账户的 IAM 角色和 EKS 容器组身份。

服务账户的 IAM 角色

服务账户的 IAM 角色（IRSA）可为 AWS 上运行的 Kubernetes 应用程序配置细粒度 IAM 权限，使其能够访问 Amazon S3 存储桶、Amazon DynamoDB 表等各种其他 AWS 资源。您可以在同一 Amazon EKS 集群中同时运行多个应用程序，并确保每个应用程序仅拥有所需的最低权限集。IRSA 旨在支持 AWS 支持的各种 Kubernetes 部署选项，例如 Amazon EKS、Amazon EKS Anywhere、AWS 云端 Red Hat OpenShift 服务，以及 Amazon EC2 实例上的自我管理 Kubernetes 集群。因此，IRSA 是使用基础 AWS 服务（如 IAM）构建的，不直接依赖 Amazon EKS 服务和 EKS API。有关更多信息，请参阅 服务账户的 IAM 角色。

EKS 容器组身份

EKS 容器组身份为集群管理员提供了一个简化的工作流，用于对应用程序进行身份验证，以访问各种其他 AWS 资源，例如 Amazon S3 存储桶、Amazon DynamoDB 表等。EKS 容器组身份仅适用于 EKS，因此简化了集群管理员配置 Kubernetes 应用程序以获取 IAM 权限的方式。现在，可以直接通过 AWS Management Console、EKS API 和 AWS CLI，用更少的步骤即可轻松配置这些权限，而且无需对集群内的任何 Kubernetes 对象执行任何操作。集群管理员无需在 EKS 和 IAM 服务之间切换，也无需使用特权 IAM 操作来配置应用程序所需的权限。现在可以跨多个集群使用 IAM 角色，无需在创建新集群时更新角色信任策略。EKS 容器组身份提供的 IAM 凭证包括角色会话标签，以及集群名称、命名空间、服务账户名称等属性。角色会话标签允许管理员根据匹配的标签访问 AWS 资源，从而使管理员能够创建可跨服务账户使用的单一角色。有关更多信息，请参阅 EKS 容器组身份。

比较 EKS 容器组身份和 IRSA

总的来说，EKS 容器组身份和 IRSA 都允许您向在 Kubernetes 集群上运行的应用程序授予 IAM 权限。但是，二者在配置方式、支持的限制和启用的功能方面有根本不同。下面，我们将比较这两种解决方案的一些关键方面。

	EKS 容器组身份	IRSA
角色可扩展性	您必须对每个角色设置一次，才能与新引入的 Amazon EKS 服务主体 `pods.eks.amazonaws.com` 建立信任。完成此一次性步骤后，每次在新集群中使用角色时，都无需更新该角色的信任策略。	每次要在新集群中使用 IAM 角色时，都必须使用新的 EKS 集群 OIDC 提供商端点，来更新该角色的信任策略。

	EKS 容器组身份	IRSA
集群可扩展性	EKS 容器组身份不需要用户设置 IAM OIDC 提供商，因此这一限制并不适用。	每个 EKS 集群都有一个与其关联的 OpenID Connect（OIDC）发布者 URL。要使用 IRSA，需要在 IAM 中为每个 EKS 集群创建一个唯一的 OpenID Connect 提供商。IAM 对每个 AWS 账户 的默认全局限制为 100 个 OIDC 提供商。如果您计划为每个使用 IRSA 的 AWS 账户 设置超过 100 个 EKS 集群，那么您将达到 IAM OIDC 提供商限制。
角色可扩展性	EKS 容器组身份不要求用户在信任策略中定义 IAM 角色和服务账户之间的信任关系，因此这一限制并不适用。	在 IRSA 中，您可以在角色的信任策略中定义 IAM 角色和服务账户之间的信任关系。默认情况下，信任策略大小的长度为 2048。这意味着您通常可以在单个信任策略中定义 4 个信任关系。虽然您可以增加信任策略长度限制，但单个信任策略中通常最多只能有 8 个信任关系。

	EKS 容器组身份	IRSA
角色可重用性	EKS 容器组身份提供的 AWS STS 临时凭证包括角色会话标签，例如集群名称、命名空间、服务账户名称。角色会话标签使管理员能够创建单个 IAM 角色，该角色可用于具有不同有效权限的多个服务账户，方法是允许根据附加到资源的 AWS 标签来访问资源，也称为基于属性的访问权限控制（ABAC）。有关更多信息，请参阅 定义 EKS 容器组身份的权限以根据标签代入角色。	不支持 AWS STS 会话标签。您可以在集群之间重用角色，但每个容器组都会获得该角色的所有权限。
支持的环境	EKS 容器组身份仅在 Amazon EKS 上可用。	IRSA 可用于 Amazon EKS、Amazon EKS Anywhere、AWS 云端 Red Hat OpenShift 服务，以及 Amazon EC2 实例上的自我管理 Kubernetes 集群。
支持的 EKS 版本	EKS Kubernetes 版本 `1.24` 或更高版本。有关特定平台版本，请参阅 EKS 容器组身份集群版本。	所有受支持的 EKS 集群版本。

EKS 容器组身份

Pod 的容器中的应用程序可以使用 AWS 开发工具包或 AWS CLI，以向使用 AWS Identity and Access Management（IAM）权限的 AWS 服务 发出 API 请求。应用程序必须通过 AWS 凭证签署 AWS API 请求。

Amazon EKS 容器组身份提供管理应用程序凭证的功能，类似于 Amazon EC2 实例配置文件为 Amazon EC2 实例提供凭证的方式。您可以将 IAM 角色与 Kubernetes 服务账户关联并配置 Pods 使用服务账户，而不是创建 AWS 凭证并将其分配到容器或使用 Amazon EC2 实例的角色。

每个 EKS 容器组身份关联都将角色映射到指定集群命名空间中的服务账户。如果您在多个集群中使用相同的应用程序，则可以在每个集群中进行相同的关联，而无需修改角色的信任策略。

如果某个容器组使用具有关联的服务账户，Amazon EKS 会在容器组的容器中设置环境变量。环境变量配置 AWS SDK（包括 AWS CLI）使用 EKS 容器组身份凭证。

EKS 容器组身份的优势

EKS 容器组身份具有以下优势：

- 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 – Pod's 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。使用容器组身份时，Pod's 容器还具有分配给 Amazon EKS 节点 IAM 角色的权限，除非您阻止 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。
- 可审计性：可通过 AWS CloudTrail 进行访问和事件日志记录，帮助确保追溯性审计。

EKS 容器组身份是一种比 服务账户的 IAM 角色 更简单的方法，因为此方法不使用 OIDC 身份提供者。EKS 容器组身份具有以下增强功能：

- 独立操作：在许多组织中，不同的团队负责创建 OIDC 身份提供者，而不是管理 Kubernetes 集群。EKS 容器组身份有明确的职责分工，EKS 容器组身份关联的所有配置都在 Amazon EKS 中完成，而 IAM 权限的所有配置都在 IAM 中完成。
- 可重用性：EKS 容器组身份使用单个 IAM 主体，而不是服务账户 IAM 角色使用的每个集群的单独主体。IAM 管理员将以下主体添加到任何角色的信任策略中，使其可供 EKS 容器组身份使用。

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            }
```

- 可扩展性：每组临时凭证均由 EKS 容器组身份中的 EKS Auth 服务使用，而不是由每个容器组中运行的每个 AWS SDK 使用。然后，在每个节点上运行的 Amazon EKS 容器组身份代理向 SDK 发放凭证。因此，每个节点的负载减少至一次，而且不会在每个容器组中重复。有关该过程的更多详细信息，请参阅 EKS 容器组身份的工作原理。

有关比较两种替代方案的更多信息，请参阅 使用 Kubernetes 服务账户授予 Kubernetes 工作负载访问 AWS 的权限。

EKS 容器组身份设置概述

完成以下过程，打开 EKS 容器组身份：

1. 设置 Amazon EKS 容器组身份代理 – 对于每个集群，您只需完成一次此步骤。
2. 配置 Kubernetes 服务账户以使用 EKS 容器组身份分派 IAM 角色 – 针对您希望应用程序拥有的每组唯一权限完成此步骤。
3. 配置 Pods 以使用 Kubernetes 服务账户 – 为需要访问 AWS 服务 的每个 Pod 完成此步骤。
4. 使用支持的 AWS 开发工具包：确认工作负载使用支持版本的 AWS SDK，并且工作负载使用默认凭证链。

EKS 容器组身份注意事项

- 您可以将一个 IAM 角色关联到每个集群中的每个 Kubernetes 服务账户。您可以编辑 EKS 容器组身份关联，来更改映射到服务账户的角色。
- 只能关联与集群属于同一 AWS 账户 的角色。您可以将其他账户访问权限委派给此账户中的角色，也即您配置的供 EKS 容器组身份使用的角色。有关关于委派访问和 `AssumeRole` 的教程，请参阅《IAM 用户指南》中的使用 IAM 角色委派 AWS 账户的访问。
- EKS 容器组身份代理是必需的。此代理作为 Kubernetes `DaemonSet` 在您的节点上运行，并且仅向其运行节点上的容器组提供凭证。有关 EKS 容器组身份代理兼容性的更多信息，请参阅以下部分 EKS 容器组身份限制：
- EKS 容器组身份代理使用节点的 `hostNetwork`，并使用节点上链路本地地址上的端口 `80` 和端口 `2703`。对于 IPv4，该地址是 `169.254.170.23`；对于 IPv6 集群，该地址是 `[fd00:ec2::23]`。

  如果禁用 `IPv6` 地址或以其他方式阻止本地主机 `IPv6` IP 地址，代理将无法启动。要在无法使用 `IPv6` 的节点上启动代理，请按照 在 EKS 容器组身份代理中禁用 IPv6 中的步骤禁用 `IPv6` 配置。

EKS 容器组身份集群版本

要使用 EKS 容器组身份，集群的平台版本必须与下表中所列的版本相同或更高，或者 Kubernetes 版本必须比下表中所列的版本更高。

Kubernetes 版本	平台版本
1.30	eks.2
1.29	eks.1
1.28	eks.4
1.27	eks.8
1.26	eks.9
1.25	eks.10
1.24	eks.13

与 EKS 容器组身份兼容的附加组件版本

> **Important**
>
> 要将 EKS 容器组身份与 EKS 附加组件结合使用，必须手动创建 EKS 容器组身份关联。请勿在 AWS Management Console 的附加组件配置中选择 IAM 角色，该角色仅用于 IRSA。

需要 IAM 凭证的 Amazon EKS 附加组件以及自行管理的附加组件可以使用 EKS 容器组身份、IRSA 或实例角色。使用支持 EKS 容器组身份的 IAM 凭证的附加组件列表包括：

- Amazon VPC CNI plugin for Kubernetes `1.15.5-eksbuild.1` 或更高版本
- AWS Load Balancer Controller `2.7.0` 或更高版本。请注意，AWS Load Balancer Controller 不能作为 EKS 附加组件使用，但可以作为自行管理的附加组件使用。

EKS 容器组身份限制

EKS 容器组身份适用于：

- 上一个主题 EKS 容器组身份集群版本 中列出的 Amazon EKS 集群版本。
- 集群中属于 Linux Amazon EC2 实例的 Worker 节点。

EKS 容器组身份不适用于：

- 中国区域
- AWS GovCloud (US).
- AWS Outposts.
- Amazon EKS Anywhere。
- 您在 Amazon EC2 上创建和运行的 Kubernetes 集群。EKS 容器组身份组件仅在 Amazon EKS 上可用。

您不能将 EKS 容器组身份用于：

- 在 Linux Amazon EC2 实例之外的任何位置运行的容器组。不支持在 AWS Fargate (Fargate) 上运行的 Linux 和 Windows 容器组。不支持 Windows Amazon EC2 实例上运行的容器组。
- 需要 IAM 凭证的 Amazon EKS 插件。EKS 插件只能使用服务账户的 IAM 角色。使用 IAM 凭证的 EKS 插件列表包括：
  - CSI 存储驱动程序：EBS CSI、EFS CSI、适用于 Lustre CSI 驱动程序的 Amazon FSx、适用于 NetApp ONTAP CSI 驱动程序的 Amazon FSx、适用于 OpenZFS CSI 驱动程序的 Amazon FSx、适用于 Kubernetes Secrets Store CSI 驱动程序的 AWS Secrets and Configuration Provider（ASCP）

> **Note**
>
> 如果这些控制器、驱动程序和插件作为自我管理的插件而不是作为 EKS 插件安装，则只要其更新为使用最新的 AWS SDK，则支持 EKS 容器组身份。

EKS 容器组身份的工作原理

Amazon EKS 容器组身份关联提供管理应用程序凭证的功能，类似于 Amazon EC2 实例配置文件为 Amazon EC2 实例提供凭证的方式。

Amazon EKS 容器组身份通过额外的 EKS Auth API，以及在每个节点上运行的代理容器组为您的工作负载提供凭证。

在您的插件中，例如，Amazon EKS 插件和自我管理控制器、运算符和其他插件，创建者需要更新软件才能使用最新的 AWS SDK。有关 EKS 容器组身份与 Amazon EKS 插件之间的兼容性列表，请参阅上一节 EKS 容器组身份限制。

在代码中使用 EKS 容器组身份

在您的代码中，您可以使用 AWS SDK 访问 AWS 服务。您可以编写代码，为使用 SDK 的 AWS 服务创建客户端，默认情况下，SDK 会在一系列位置中搜索要使用的 AWS Identity and Access Management 凭证。找到有效凭证后，搜索停止。有关使用的默认位置的更多信息，请参阅《AWS SDK 和工具参考指南》中的凭证提供程序链。

EKS 容器组身份已添加到容器凭证提供程序，可在默认凭证链的一个步骤中搜索。如果您的工作负载当前使用凭证链中较早的证书，则即使您为同一工作负载配置了 EKS 容器组身份关联，这些凭证也将继续使用。这样，在删除旧凭证之前，您可以先创建关联，从而安全地从其他类型的凭证迁移。

容器凭证提供程序从每个节点上运行的代理处提供临时凭证。在 Amazon EKS 中，代理是 Amazon EKS 容器组身份，而在 Amazon Elastic Container Service 中，代理则是 `amazon-ecs-agent`。SDK 使用环境变量来定位要连接的代理。

相比之下，服务账户的 IAM 角色提供了一个 Web 身份令牌，AWS SDK 必须使用 `AssumeRoleWithWebIdentity` 与 AWS Security Token Service 进行交换。

EKS 容器组身份代理如何使用 Pod

1. 当 Amazon EKS 启动一个新的容器组，而该容器组使用与 EKS 容器组身份关联的服务账户时，集群将添加以下内容到 Pod 清单中：

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes 选择要在哪个节点上运行容器组。然后，节点上的 Amazon EKS 容器组身份代理使用 AssumeRoleForPodIdentity 操作从 EKS Auth API 检索临时凭证。
3. EKS 容器组身份代理可为在容器中运行的 AWS SDK 提供这些凭证。
4. 在应用程序中使用 SDK 时，无需指定凭证提供程序使用默认凭证链。或者，您可以指定容器凭证提供程序。有关使用的默认位置的更多信息，请参阅《AWS SDK 和工具参考指南》中的凭证提供程序链。
5. SDK 使用环境变量连接到 EKS 容器组身份代理并检索凭证。

> **Note**
>
> 如果您的工作负载当前使用凭证链中较早的证书，则即使您为同一工作负载配置了 EKS 容器组身份关联，这些凭证也将继续使用。

设置 Amazon EKS 容器组身份代理

Amazon EKS 容器组身份关联提供管理应用程序凭证的功能，类似于 Amazon EC2 实例配置文件为 Amazon EC2 实例提供凭证的方式。

Amazon EKS 容器组身份通过额外的 EKS Auth API，以及在每个节点上运行的代理容器组为您的工作负载提供凭证。

注意事项

- **IPv6**

  默认情况下，EKS 容器组身份代理会侦听 IPv4 和 IPv6 地址，以便为容器组请求凭证。对于 IPv4，代理会使用环回（本地主机）IP 地址 169.254.170.23，对于 IPv6，则会使用本地主机 IP 地址 [fd00:ec2::23]。

  如果禁用 IPv6 地址或以其他方式阻止本地主机 IPv6 IP 地址，代理将无法启动。要在无法使用 IPv6 的节点上启动代理，请按照 在 EKS 容器组身份代理中禁用 IPv6 中的步骤禁用 IPv6 配置。

创建 Amazon EKS 容器组身份代理

代理先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。集群版本和平台版本必须与 EKS 容器组身份集群版本 中列出的版本相同或更高。

- 节点角色有权让代理在 EKS Auth API 中执行 `AssumeRoleForPodIdentity` 操作。您可以使用 AWS托管策略：AmazonEKSWorkerNodePolicy 或添加自定义策略，自定义策略与以下策略类似：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

  可通过标签来限制此操作，以限制使用代理的容器组可以分派哪些角色。

- 节点可以访问 Amazon ECR 并从中下载映像。插件的容器映像位于 Amazon 容器镜像注册表 中列出的注册表中。

  请注意，您可以在 AWS Management Console 中的可选配置设置，以及 AWS CLI 中的 `--configuration-values` 更改映像位置并为 EKS 插件提供 `imagePullSecrets`。

- 节点可以访问 Amazon EKS Auth API。对于私有集群，AWS PrivateLink 中的 `eks-auth` 端点是必需的。

AWS Management Console

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，然后为您要配置 EKS 容器组身份代理插件的集群选择集群名称。
3. 选择附加组件选项卡。
4. 选择获取更多附加组件。
5. 选择 EKS 容器组身份代理插件框右上角的框，然后选择下一步。
6. 在配置选定插件设置页面上，从版本下拉列表中选择任意版本。
7. （可选）展开可选配置设置以输入其他配置。例如，您可以提供备用容器映像位置和 `ImagePullSecrets`。带有已接受键的 JSON Schema 显示在插件配置架构中。

在配置值中输入配置键和值。

8. 选择下一步。
9. 确认 EKS 容器组身份代理容器组正在您的集群上运行。

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

示例输出如下。

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1
 Running   1 (24h ago)   24h
```

现在，您可以在集群中使用 EKS 容器组身份关联。有关更多信息，请参阅 [配置 Kubernetes 服务账户以使用 EKS 容器组身份分派 IAM 角色]。

AWS CLI

1. 运行以下 AWS CLI 命令：将 `my-cluster` 替换为您的集群名称。

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-
agent --addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> EKS 容器组身份代理不将 `service-account-role-arn` 用于服务账户的 IAM 角色。您必须为 EKS 容器组身份代理提供节点角色的权限。

2. 确认 EKS 容器组身份代理容器组正在您的集群上运行。

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

示例输出如下。

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
```

```
eks-pod-identity-agent-prnsh                                    1/1
 Running   1 (24h ago)   24h
```

现在，您可以在集群中使用 EKS 容器组身份关联。有关更多信息，请参阅 配置 Kubernetes 服务账户以使用 EKS 容器组身份分派 IAM 角色。

更新 Amazon EKS 容器组身份代理

更新 Amazon EKS 类型的附加组件。如果您尚未将 Amazon EKS 类型的插件添加到集群，请参阅 创建 Amazon EKS 容器组身份代理。

AWS Management Console

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，然后为您要配置 EKS 容器组身份代理插件的集群选择集群名称。
3. 选择附加组件选项卡。
4. 如果有新版本的插件可用，EKS 容器组身份代理会有一个更新版本按钮。选择更新版本。
5. 在配置 Amazon EKS 容器组身份代理页面上，从版本下拉列表中选择新版本。
6. 选择保存更改。

   可能需要几秒钟才能完成更新。然后，通过查看状态确认插件版本已更新。

AWS CLI

1. 查看集群上当前安装的附加组件版本。将 *my-cluster* 替换为您的集群名称。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --query "addon.addonVersion" --output text
   ```

   示例输出如下。

   ```
   v1.0.0-eksbuild.1
   ```

   您需要先创建附加组件，然后才能使用此过程对其进行更新。

2. 使用 AWS CLI 更新您的附加组件。如果您想要使用 AWS Management Console 或 `eksctl` 更新附加组件，则请参阅 更新附加组件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令。

   - 将 *my-cluster* 替换为您的集群名称。
   - 将 *v1.0.0-eksbuild.1* 替换为所需的版本。
   - 请将 *111122223333* 替换为您的账户 ID。
   - 运行以下命令：

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --addon-version v1.0.0-eksbuild.1'
   ```

   可能需要几秒钟才能完成更新。

3. 确认附加组件版本已更新。将 *my-cluster* 替换为您的集群名称。

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent
   ```

   可能需要几秒钟才能完成更新。

   示例输出如下。

   ```
   {
       "addon": {
           "addonName": "eks-pod-identity-agent",
           "clusterName": "my-cluster",
           "status": "ACTIVE",
           "addonVersion": "v1.0.0-eksbuild.1",
           "health": {
               "issues": []
           },
           "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/eks-pod-identity-agent/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
           "createdAt": "2023-04-12T18:25:19.319000+00:00",
           "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
           "tags": {}
       }
   }
   ```

EKS 容器组身份代理配置

在 EKS 容器组身份代理中禁用 **IPv6**

AWS Management Console

在 AWS Management Console 中禁用 **IPv6**

1. 要在 EKS 容器组身份代理中禁用 IPv6，请将以下配置添加到 EKS 附加组件的可选配置设置中。

   a. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。

   b. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其配置附加组件的集群的名称。

   c. 选择附加组件选项卡。

   d. 选择 EKS 容器组身份代理附加组件框右上角的框，然后选择编辑。

   e. 在配置 EKS 容器组身份代理页面上：

      i. 选择您想使用的 Version（版本）。我们建议您保留与上一步相同的版本，并通过单独的操作更新版本和配置。

      ii. 展开可选配置设置。

      iii. 在配置值中输入 JSON 键 `"agent":`，以及键为 `"additionalArgs":` 的嵌套 JSON 对象的值。生成的文本必须是有效的 JSON 对象。如果此键和值是文本框中的唯一数据，请用大括号 `{}` 将键和值括起来。以下示例显示网络策略已启用：

      ```
      {
          "agent": {
              "additionalArgs": {
                  "-b": "169.254.170.23"
              }
          }
      }
      ```

      此配置会将 IPv4 地址设置为代理所使用的唯一地址。

   f. 要通过替换 EKS 容器组身份代理容器组来应用新配置，请选择保存更改。

   Amazon EKS 通过推出适用于 EKS 容器组身份代理的 Kubernetes `DaemonSet`，将更改应用到 EKS 附加组件。可在 AWS Management Console 中的附加组件更新历史记录中跟

踪推出状态和 `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`。

`kubectl rollout` 具有以下命令：

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推出时间过长，Amazon EKS 将撤销推出，并且会将类型为附加组件更新且状态为失败的消息添加到附加组件的更新历史记录中。要调查任何问题，请先查看推出历史记录，然后在 EKS 容器组身份代理容器组上运行 `kubectl logs`，进而查看 EKS 容器组身份代理的日志。

2. 如果更新历史记录中的新条目状态为成功，则表示推出已完成，并且附加组件正在所有 EKS 容器组身份代理容器组中使用新配置。

AWS CLI

在 AWS CLI 中禁用 **IPv6**

- 要在 EKS 容器组身份代理中禁用 IPv6，请将以下配置添加到 EKS 附加组件的配置值中。

  运行以下 AWS CLI 命令：将 `my-cluster` 替换为集群的名称，并将 IAM 角色 ARN 替换为您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-
agent \
    --resolve-conflicts PRESERVE --configuration-values '{"agent":
{"additionalArgs": { "-b": "169.254.170.23"}}}'
```

  此配置会将 IPv4 地址设置为代理所使用的唯一地址。

Amazon EKS 通过推出适用于 EKS 容器组身份代理的 Kubernetes DaemonSet，将更改应用到 EKS 附加组件。可在 AWS Management Console 中的附加组件更新历史记录中跟踪推出状态和 `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`。

`kubectl rollout` 具有以下命令：

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推出时间过长，Amazon EKS 将撤销推出，并且会将类型为附加组件更新且状态为失败的消息添加到附加组件的更新历史记录中。要调查任何问题，请先查看推出历史记录，然后在 EKS 容器组身份代理容器组上运行 `kubectl logs`，进而查看 EKS 容器组身份代理的日志。

配置 Kubernetes 服务账户以使用 EKS 容器组身份分派 IAM 角色

本主题介绍如何配置 Kubernetes 服务账户，以使用 EKS 容器组身份分派 AWS Identity and Access Management（IAM）角色。然后，配置为使用服务账户的任何 Pods 都可以访问该角色有权访问的任何 AWS 服务。

要创建 EKS 容器组身份关联，只需一步；您可以通过 AWS Management Console、AWS CLI、AWS SDK、AWS CloudFormation，以及其他工具在 EKS 中创建关联。任何 Kubernetes 对象中都没有关于集群内关联的任何数据或元数据，也不需要向服务账户帐户添加任何注释。

先决条件

- 现有集群。如果您没有，可以按照 开始使用 Amazon EKS 指南之一创建一个。
- 创建关联的 IAM 主体必须具有 `iam:PassRole`。
- 在您的设备或 AWS CloudShell 上安装和配置的最新版本 AWS CLI。您可以使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1` 检查您的当前版本。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，

请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI 和使用 aws configure 进行快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。要对其进行更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到您的主目录。

- 您的设备或 AWS 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 1.29，则可以将 kubectl 的 1.28、1.29 或 1.30 版本与之配合使用。要安装或升级 kubectl，请参阅 安装或更新 kubectl。
- 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 集群创建或更新 kubeconfig 文件。

创建 EKS 容器组身份关联

AWS Management Console

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，然后为您要配置 EKS 容器组身份代理插件的集群选择集群名称。
3. 选择访问选项卡。
4. 在容器组身份关联中，选择创建。
5. 对于 IAM 角色，选择具有工作负载所需权限的 IAM 角色。

> **Note**
>
> 该列表仅包含具有以下信任策略的角色，该策略允许 EKS 容器组身份使用这些角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
```

```
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

在将临时凭证传递给您的容器组之前，EKS 容器组身份使用 `AssumeRole` 分派 IAM 角色。

`sts:TagSession`

EKS 容器组身份使用 `TagSession` 在对 AWS STS 的请求中包含会话标签。

您可以在信任策略的 condition keys 中使用这些标签，来限制哪些服务账户、命名空间和集群可以使用此角色。

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 定义的条件。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。

6. 对于 Kubernetes 命名空间，选择包含服务账户和工作负载的 Kubernetes 命名空间。或者，您可以按名称指定集群中不存在的命名空间。
7. 对于 Kubernetes 服务账户，选择要使用的 Kubernetes 服务账户。Kubernetes 工作负载清单必须指定此服务账户。或者，您可以按名称指定集群中不存在的服务账户。
8. （可选）对于标签，选择添加标签以在键值对中添加元数据。这些标签将应用于关联，并可在 IAM 策略中使用。

   您可以重复此步骤以添加多个标签。
9. 选择创建。

AWS CLI

1. 要将现有 IAM 策略关联到您的 IAM 角色，请跳至下一步。

创建一个 IAM 策略。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 AWS 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 策略。

a. 创建一个包含 Pods 访问 AWS 服务 所需权限的文件。有关所有 AWS 服务 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 策略。

```
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。

1. 如果您有要分派 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
```

运行以下命令。

```
kubectl apply -f my-service-account.yaml
```

2. 运行以下命令为 IAM 角色创建信任策略文件。

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

3. 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"
```

4. 将 IAM 策略附加到您的角色。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::111122223333:policy/my-policy
```

> **Note**
>
> 与服务账户的 IAM 角色不同，EKS 容器组身份不在服务账户上使用注释。

5. 运行以下命令以创建关联。将 `my-cluster` 替换为集群的名称，将 *`my-service-account`* 替换为您所需的名称，并根据需要将 *`default`* 替换为其他命名空间。

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-
arn arn:aws:iam::111122223333:role/my-role --namespace default --service-
account my-service-account
```

示例输出如下。

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws:iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> 您可以按名称指定集群中不存在的命名空间和服务账户。您必须创建命名空间、服务账户，以及使用服务账户的工作负载，EKS 容器组身份关联才能正常工作。

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
```

示例输出如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow EKS Auth service to assume this role for Pod
 Identities",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

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
```

示例输出如下。

```
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
```

示例输出如下。

```
{
    "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
```

示例输出如下。

```
{
    "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。

后续步骤

[配置 Pods 以使用 Kubernetes 服务账户]

配置 Pods 以使用 Kubernetes 服务账户

如果 Pod 需要访问 AWS 服务，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 AWS 服务 的 AWS 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 指南之一创建一个。
- 现有 Kubernetes 服务账户和 EKS 容器组身份关联（将服务账户与 IAM 角色关联）。该角色必须具有关联的 IAM 策略，其中包含您希望您的 Pods 必须具有的权限，以便使用 AWS 服务。有关如何创建和配置服务账户和角色的更多信息，请参阅 配置 Kubernetes 服务账户以使用 EKS 容器组身份分派 IAM 角色。
- 在您的设备或 AWS CloudShell 上安装和配置的最新版本 AWS CLI。您可以使用 `aws --version | cut -d / -f2 | cut -d ' ' -f1` 检查您的当前版本。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI 和使用 aws configure 进行快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。要对其进行更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到您的主目录。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 kubectl。
- 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 集群创建或更新 kubeconfig 文件。

要配置 Pod 以使用服务账户

1. 使用以下命令创建部署清单，您可以使用该部署清单部署 Pod 以确认配置。将 *example values* 替换为您自己的值。

```
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:X.XX
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
   ```

   示例输出如下。

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. 确认 Pod 已挂载服务账户令牌文件。

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
    AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
   ```

   示例输出如下。

   ```
   AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
   pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
   ```

4. 确认您的 Pods 可以使用您在附加到角色的 IAM 策略中分配的权限与 AWS 服务 进行交互。

> **Note**
>
> 当 Pod 使用来自与服务账户关联的 IAM 角色的 AWS 凭证时，该 Pod 容器中的 AWS 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 Pods 使用的 AWS SDK 版本支持通过 EKS 容器组身份关联分派 IAM 角色。有关更多信息，请参阅 使用支持的 AWS 开发工具包。

b. 确认部署正在使用服务账户。

```
kubectl describe deployment my-app | grep "Service Account"
```

示例输出如下。

```
Service Account:  my-service-account
```

定义 EKS 容器组身份的权限以根据标签代入角色

EKS 容器组身份将标签附加到每个容器组的临时凭证上，其中包含集群名称、命名空间、服务账户名称等属性。这些角色会话标签允许管理员根据匹配的标签访问 AWS 资源，从而使管理员能够创建可跨服务账户使用的单一角色。通过添加对角色会话标签的支持，客户可以在集群之间和集群内的工作负载之间实施更严格的安全边界，同时重复使用相同的 IAM 角色和 IAM 策略。

例如，如果对象标记具有 EKS 集群的名称，则以下策略允许执行 `s3:GetObject` 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

EKS 容器组身份添加的会话标签列表

以下列表包含添加到 AssumeRole 请求（由 Amazon EKS 发出）的所有标签键。要在策略中使用这些标签，请使用 ${aws:PrincipalTag/ 后跟键，例如 ${aws:PrincipalTag/kubernetes-namespace}。

- eks-cluster-arn
- eks-cluster-name
- kubernetes-namespace
- kubernetes-service-account
- kubernetes-pod-name
- kubernetes-pod-uid

跨账户标签

EKS 容器组身份添加的所有会话标签都是可传递的；将标签键和值传递到任何 AssumeRole 操作，以便您的工作负载用于将角色切换到其他账户。您可以在其他账户的策略中使用这些标签，来限制跨账户场景中的访问。有关更多信息，请参阅《IAM 用户指南》中的使用会话标签链接角色。

自定义标签

EKS 容器组身份无法在其执行的 AssumeRole 操作中添加其他自定义标签。但是，您应用于 IAM 角色的标签始终可用，且格式相同：${aws:PrincipalTag/ 后跟键，例如 ${aws:PrincipalTag/MyCustomTag}。

> **Note**
>
> 如果发生冲突，通过 sts:AssumeRole 请求添加到会话中的标签优先。例如，假设 EKS 代入客户角色时，Amazon EKS 向会话添加键 eks-cluster-name 和值 my-cluster。您还

向 IAM 角色添加了一个值为 `my-own-cluster` 的 `eks-cluster-name` 标签。在这种情况下，前者优先，`eks-cluster-name` 标签的值为 `my-cluster`。

使用支持的 AWS 开发工具包

Important

该文档的早期版本不正确。适用于 Java 的 AWS SDK v1 不支持 EKS 容器组身份。

在使用 EKS 容器组身份 时，Pods 中的容器必须使用支持从 EKS 容器组身份代理分派 IAM 角色的 AWS SDK 版本。请确保为您的 AWS 开发工具包使用以下版本或更高版本：

- Java（版本 2）：2.21.30
- Go v1：v1.47.11
- Go v2：release-2023-11-14
- Python（Boto3）– 1.34.41
- Python (botocore) – 1.34.41
- AWS CLI：1.30.0

  AWS CLI – 2.15.0
- JavaScript v2 – 2.1550.0
- JavaScript v3 – v3.458.0
- Kotlin – v1.0.1
- Ruby：3.188.0
- Rust – release-2024-03-13
- C++ – 1.11.263
- .NET – 3.7.734.0
- PowerShell – 4.1.502
- PHP：3.287.1

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 AWS 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。

有关支持 EKS 容器组身份的附加组件列表，请参阅 与 EKS 容器组身份兼容的附加组件版本。

使用 EKS 容器组身份凭证

要使用来自 EKS 容器组身份关联的凭证，您的代码可以使用任何 AWS SDK 为具有 SDK 的 AWS 服务创建客户端，默认情况下，SDK 会在一系列位置中搜索要使用的 AWS Identity and Access Management 凭证。如果您在创建客户端或者初始化 SDK 时未指定凭证提供程序，则将使用 EKS 容器组身份凭证。

由于 EKS 容器组身份已添加到容器凭证提供程序，可在默认凭证链的一个步骤中搜索，因此该操作可行。如果您的工作负载当前使用凭证链中较早的证书，则即使您为同一工作负载配置了 EKS 容器组身份关联，这些凭证也将继续使用。

有关 EKS 容器组身份如何工作的更多信息，请参阅 EKS 容器组身份的工作原理。

EKS 容器组身份角色

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

`sts:AssumeRole`

在将临时凭证传递给您的容器组之前，EKS 容器组身份使用 `AssumeRole` 分派 IAM 角色。

`sts:TagSession`

EKS 容器组身份使用 `TagSession` 在对 AWS STS 的请求中包含会话标签。

您可以在信任策略的 condition keys 中使用这些标签，来限制哪些服务账户、命名空间和集群可以使用此角色。

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 定义的条件。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。

服务账户的 IAM 角色

Pod 的容器中的应用程序可以使用 AWS 开发工具包或 AWS CLI，以向使用 AWS Identity and Access Management（IAM）权限的 AWS 服务 发出 API 请求。应用程序必须通过 AWS 凭证签署 AWS 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 Pods 使用服务账户，而不是创建 AWS 凭证并将其分配到容器或使用 Amazon EC2 实例的角色。您不能将 IAM 角色用于包含 AWS Outposts 上的 Amazon EKS 的本地集群的服务账户。

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 – Pod's 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 角色的权限，除非您阻止 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 AWS 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. 为集群创建 IAM OIDC 提供商 – 对于每个集群，您只需完成一次此步骤。

> **Note**
>
> 如果您启用 EKS VPC 端点，则无法从该 VPC 内部访问 EKS OIDC 服务端点。因此，您在 VPC 中使用 `eksctl` 创建 OIDC 提供商等操作将不起作用，并且在尝试请求 `https://oidc.eks.region.amazonaws.com` 时将导致超时。错误消息示例如下：
>
> ```
> ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
> ```

要完成此步骤，您可以在 VPC 外部运行该命令，例如在 AWS CloudShell 中或在连接到互联网的计算机上。

2. 配置 Kubernetes 服务账户来代入 IAM 角色 – 针对您希望应用程序拥有的每组唯一权限完成此步骤。
3. 配置 Pods 以使用 Kubernetes 服务账户 – 为需要访问 AWS 服务 的每个 Pod 完成此步骤。
4. 使用支持的 AWS 开发工具包：确认工作负载使用支持版本的 AWS SDK，并且工作负载使用默认凭证链。

IAM、Kubernetes 以及 OpenID Connect（OIDC）背景信息

2014 年，AWS 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 AWS 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 AWS 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 AWS 服务 交互，包括 Amazon S3 和 DynamoDB。

每个 JWT 令牌均由签名密钥对签名。密钥由 Amazon EKS 管理的 OIDC 提供商提供，私钥每 7 天轮换一次。Amazon EKS 会保留公钥直至其过期。如果您连接外部 OIDC 客户端，请注意您需要在公钥过期之前刷新签名密钥。了解如何the section called “获取签名密钥”。

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 AWS Identity and Access Management（IAM）角色用于服务账户，您的集群 OIDC 发布者 URL 必须存在 IAM OIDC 提供商。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。

- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 您的设备或 AWS 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 1.29，则可以将 kubectl 的 1.28、1.29 或 1.30 版本与之配合使用。要安装或升级 kubectl，请参阅 安装或更新 kubectl。
- 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 集群创建或更新 kubeconfig 文件。

您可以使用 eksctl 或AWS Management Console为集群创建 IAM OIDC 提供商。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。

使用 **eksctl** 为集群创建 IAM OIDC 身份提供商

1. 确定您的集群的 OIDC 发布者 ID。

   检索集群的 OIDC 发布者 ID 并将其存储在变量中。将 *my-cluster* 替换为您自己的值。

   ```
   cluster_name=my-cluster
   ```

   ```
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   ```

   ```
   echo $oidc_id
   ```

2. 确定您的账户中是否已存在具有您的集群发布者 ID 的 IAM OIDC 提供商。

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果返回了输出，则表示您的集群已经有 IAM OIDC 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。

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
```

> **Note**
>
> 如果您启用 EKS VPC 端点，则无法从该 VPC 内部访问 EKS OIDC 服务端点。因此，您在 VPC 中使用 `eksctl` 创建 OIDC 提供商等操作将不起作用，并且在尝试请求 `https://oidc.eks.region.amazonaws.com` 时将导致超时。错误消息示例如下：
>
> ```
> ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
> ```
>
> 要完成此步骤，您可以在 VPC 外部运行该命令，例如在 AWS CloudShell 中或在连接到互联网的计算机上。

AWS Management Console

使用 AWS 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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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）。

后续步骤

配置 Kubernetes 服务账户来代入 IAM 角色

配置 Kubernetes 服务账户来代入 IAM 角色

本主题介绍如何配置 Kubernetes 服务账户以分派 AWS Identity and Access Management（IAM）角色。然后，配置为使用服务账户的任何 Pods 都可以访问该角色有权访问的任何 AWS 服务。

先决条件

- 现有集群。如果您没有，可以按照 开始使用 Amazon EKS 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（IAM）角色或如何创建一个（IAM）角色，请参阅 为集群创建 IAM OIDC 提供商。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 您的设备或 AWS 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 1.29，则可以将 kubectl 的 1.28、1.29 或 1.30 版本与之配合使用。要安装或升级 kubectl，请参阅 安装或更新 kubectl。
- 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 集群创建或更新 kubeconfig 文件。

将 IAM 角色与 Kubernetes 服务账户关联。

1. 要将现有 IAM 策略关联到您的 IAM 角色，请跳至下一步。

创建一个 IAM 策略。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 AWS 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 策略。

a. 创建一个包含 Pods 访问 AWS 服务 所需权限的文件。有关所有 AWS 服务 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 策略。

```
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 AWS CLI。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

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**。

AWS CLI

1. 如果您有要分派 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
   ```

2. 使用以下命令将 AWS 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 --region
 $AWS_REGION --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* 替换为要分派角色的 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分派角色。要允许其他 AWS 账户 中的角色（而不是您的集群所在账户中的角色）分派该角色户，请参阅 跨账户 IAM 权限 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 策略附加到您的角色。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分派的 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 AWS 账户 中的角色（而不是您的集群所在账户中的角色）分派该角色户。那么，请确保指定其他账户中的 AWS 账户 和角色。有关更多信息，请参阅 跨账户 IAM 权限。

```
kubectl annotate serviceaccount -n $namespace $service_account
 eks.amazonaws.com/role-arn=arn:aws:iam::$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
```

示例输出如下。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-
service-account",
```

```
                    "oidc.eks.region-code.amazonaws.com/
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
```

示例输出如下。

```
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
```

示例输出如下。

```
{
    "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
```

示例输出如下。

```
{
    "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
```

示例输出如下。

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/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）为服务账户配置 AWS Security Token Service 端点。AWS 建议使用区域 AWS 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。

后续步骤

配置 Pods 以使用 Kubernetes 服务账户

配置 Pods 以使用 Kubernetes 服务账户

如果 Pod 需要访问 AWS 服务，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 AWS 服务 的 AWS 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 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（IAM）角色或如何创建一个（IAM）角色，请参阅 为集群创建 IAM OIDC 提供商。
- 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 策略，其中包含您希望您的 Pods 必须具有的权限，以便使用 AWS 服务。有关如何创建和配置服务账户和角色的更多信息，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。
- 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 集群创建或更新 `kubeconfig` 文件。

要配置 Pod 以使用服务账户

1. 使用以下命令创建部署清单，您可以使用该部署清单部署 Pod 以确认配置。将 *`example values`* 替换为您自己的值。

```
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:X.XX
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
   ```

   示例输出如下。

   ```
   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:
   ```

   示例输出如下。

   ```
   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 配置 Kubernetes 服务账户来代入 IAM 角色。

   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:
```

示例输出如下。

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
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 Pods。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。AWS 开发工具包支持的版本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 AWS 服务 进行交互。

> **Note**
>
> 当 Pod 使用来自与服务账户关联的 IAM 角色的 AWS 凭证时，该 Pod 容器中的 AWS 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 Pods 使用支持通过 OpenID Connect Web 身份令牌文件分派 IAM 角色的 AWS 开发工具包版本。有关更多信息，请参阅 使用支持的 AWS 开发工具包。

b. 确认部署正在使用服务账户。

```
kubectl describe deployment my-app | grep "Service Account"
```

示例输出如下。

```
Service Account:  my-service-account
```

c. 如果您的 Pods 仍然无法访问服务，请查看 配置 Kubernetes 服务账户来代入 IAM 角色 中描述的步骤，以确认您的角色和服务账户的配置都正确。

为服务账户配置 AWS Security Token Service 端点

如果您将 Kubernetes 服务账户与 服务账户的 IAM 角色 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 AWS 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
1.30	eks.2	区域性
1.29	eks.1	区域性
1.28	eks.1	区域性
1.27	eks.1	区域性
1.26	eks.1	区域性
1.25	eks.1	区域性
1.24	eks.2	区域性
1.23	eks.1	区域性

AWS 建议使用区域 AWS 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。AWS Security Token Service 必须在 Pod 运行的 AWS 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 AWS 区域 的服务出现故障时选择其他 AWS 区域。有关更多信息，请参阅 IAM 用户指南中的在 AWS 区域中管理 AWS STS。

先决条件

- 现有集群。如果还没有，可以使用 开始使用 Amazon EKS 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。
- 现有的 Kubernetes 服务账户配置为与适用于服务账户的 Amazon EKS IAM 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 Pod。确定 Pod 在其中运行的 AWS 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:
   ```

   示例输出如下。

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   在之前的输出中，Pod 在 AWS 区域 us-west-2 中的节点上运行。
2. 确定 Pod's 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
   ```

   示例输出如下。

   ```
   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 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.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/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 AWS STS 全局端点提出请求，且您不会覆盖在 Amazon EKS 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。

如果您使用 服务账户的 IAM 角色 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/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/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 AWS 区域 有期望，则进行必要的更改以使用适当的 AWS 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
```

示例输出如下。

```
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 提供商和配置 Kubernetes 服务账户来代入 IAM 角色中的说明，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 策略，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/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 AWS 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 AWS 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 AWS 上进行构建的工具。

使用支持的 AWS 开发工具包

在使用服务账户的 IAM 角色时，Pods 中的容器必须使用支持通过 OpenID Connect Web 身份令牌文件分派 IAM 角色的 AWS 开发工具包版本。请确保为您的 AWS 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
- AWS CLI - 1.16.232
- 节点：2.525.0 和 3.27.0
- Ruby – 3.58.0
- C++ – 1.7.174
- .NET：3.3.659.1，您必须还包括 AWSSDK.SecurityToken。
- PHP – 3.110.7

许多流行的 Kubernetes 附加组件，如 Cluster Autoscaler、AWS Load Balancer Controller 是什么？和 Amazon VPC CNI plugin for Kubernetes 都支持服务账户的 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 AWS 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。

使用凭证

要使用服务账户的 IAM 角色凭证，您的代码可以使用任何 AWS SDK 为具有 SDK 的 AWS 服务创建客户端，默认情况下，SDK 会在一系列位置中搜索要使用的 AWS Identity and Access Management 凭证。如果您在创建客户端或者初始化 SDK 时未指定凭证提供程序，则将使用服务账户凭证的 IAM 角色

由于服务账户的 IAM 角色已添加为默认凭证链中的一个步骤，因此该操作可行。如果您的工作负载当前使用凭证链中较早的证书，则即使您为同一工作负载的服务账户配置了 IAM 角色，这些凭证也将继续使用。

SDK 使用 `AssumeRoleWithWebIdentity` 操作自动将服务账户 OIDC 令牌交换为 AWS Security Token Service 中的临时凭证。Amazon EKS 和此 SDK 操作会在临时凭证到期前进行续订，从而继续轮换临时凭证。

获取签名密钥

Kubernetes 向每个 `ProjectedServiceAccountToken` Kubernetes 颁发一个 Service Account。此令牌是一个 OIDC 令牌，进一步讲是一种 JSON web token (JWT)。Amazon EKS 为包含令牌的签名密钥的每个集群托管公有 OIDC 端点，这样外部系统就可以验证。

您需要获取 OIDC 公有签名密钥（也称为 JSON Web Key Set (JWKS)）才能验证 `ProjectedServiceAccountToken`。在应用程序中使用这些密钥来验证令牌。例如，您可以使用 PyJWT Python 库来验证使用这些密钥的令牌。有关 `ProjectedServiceAccountToken` 的更多信息，请参阅 the section called “IAM、Kubernetes 以及 OpenID Connect（OIDC）背景信息”。

先决条件

- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- AWS CLI – 与 AWS 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 AWS Command Line Interface 用户指南中的安装、更新和卸载 AWS CLI。在安装 AWS CLI 后，建议您还要对其进行配置。有关更多信息，请参阅 AWS Command Line Interface 用户指南中的如何使用 `aws configure` 快速配置。

获取 OIDC 公有签名密钥 (AWS CLI)

1. 使用 AWS CLI 检索适用于 Amazon EKS 集群的 OIDC URL。

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. 使用 curl 或类似工具检索公有签名密钥。结果是 JSON Web Key Set (JWKS)。

> **Important**
>
> Amazon EKS 会限制对 OIDC 端点的调用。您应该缓存公共签名密钥。请遵守响应中包含的 `cache-control` 标题。

> **Important**
>
> Amazon EKS 每七天轮换一次 OIDC 签名密钥。

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}
```

Amazon EKS 节点

Kubernetes 节点是运行容器化应用程序的机器。每个节点包含下列组件：

- 容器运行时：负责运行容器的软件。
- **kubelet** – 确保该容器的运行状况良好且在其关联的 Pod 中运行。
- **kube-proxy** – 维护允许与您的 Pods 通信的网络规则。

有关更多信息，请参阅 Kubernetes 文档中的节点。

您的 Amazon EKS 集群可以在自行管理的节点、Amazon EKS 托管节点组和 AWS Fargate 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。

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

> **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，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	AWS Fargate
可以部署到 AWS Outposts	否	是	不支持

标准	EKS 托管节点组	自行管理的节点	AWS Fargate
可以部署到 AWS 本地区域	否	是 – 有关更多信息，请参阅 Amazon EKS 和 AWS 本地区域。	否
可以运行需要 Windows 的容器	是	是 – 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 – 仅限 Amazon Linux 节点	是 – 仅限 Amazon Linux	否
可以运行需要 GPU 的工作负载	是 – 仅限 Amazon Linux 节点	是 – 仅限 Amazon Linux	否
可以运行需要 ARM 处理器的工作负载	是	是	不支持
可以运行 AWS Bottlerocket	是	是	不支持
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
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。

标准	EKS 托管节点组	自行管理的节点	AWS Fargate
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 – 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 AWS CloudFormation 模板来部署 Linux (x86)、Linux (Arm) 或 Windows 节点。	否
必须保护、维护和修补 Amazon EC2 实例的操作系统	是	是	不支持
在部署节点时提供引导参数，例如额外的 `kubelet` 参数。	是 – 搭配使用 `eksctl` 或启动模板与自定义 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 – 结合使用启动模板和自定义 AMI。有关更多信息，请参阅 使用启动模板自定义托管节点。	是 – 有关更多信息，请参阅 容器组（pod）的自定义网络。	否

标准	EKS 托管节点组	自行管理的节点	AWS Fargate
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	是	不支持
可以将自己的自定义 CNI 部署到节点	是 – 结合使用启动模板和自定义 AMI	是	不支持
必须自己更新节点 AMI	是 – 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 - 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

标准	EKS 托管节点组	自行管理的节点	AWS Fargate
必须自己更新节点 Kubernetes 版本	是 – 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 - 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	是	不支持
可以为 Pods 使用 Amazon EFS 存储	是	是	是
可以为 Pods 使用 Amazon FSx for Lustre 存储	是	是	不支持
可以对服务使用 Network Load Balancer	是	是	是的，当使用 创建网络负载均衡器 时
Pod 可以在公有子网中运行	是	是	不支持
可以将不同的 VPC 安全组分配给各个 Pods	是 – 仅限 Linux 节点	是 – 仅限 Linux 节点	是
可以运行 Kubernetes DaemonSets	是	是	不支持
在 Pod 清单中支持 `HostPort` 和 `HostNetwork`	是	是	不支持

标准	EKS 托管节点组	自行管理的节点	AWS Fargate
AWS 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
可以在 Amazon EC2 专属主机上运行容器	是	是	不支持
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AWS 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 AWS 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`、AWS CLI、AWS API 或基础设施即代码工具（包括 AWS 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 AWS 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 AWS 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 – AWS Management Console 和 AWS CLI。

要将托管节点组添加到现有集群，请参阅 创建托管节点组。

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 AWS 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。

> **Important**
>
> 如果要使用 Kubernetes Autoscaling 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。例如，您可以指定额外的 `kubelet` 参数并使用自定义 AMI。有关更多信息，请参阅 使用启动模板自定义托管节点。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。
- 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 AWS CloudFormation 模板并在 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。
- 在私有子网中部署托管节点组时，必须确保它可以访问 Amazon ECR 来拉取容器映像。您可以通过将 NAT 网关连接到子网的路由表或添加以下 AWS PrivateLink VPC 端点来实现此目的：

  - Amazon ECR API 端点接口 – com.amazonaws.*region-code*.ecr.api
  - Amazon ECR Docker 注册表 API 端点接口 – com.amazonaws.*region-code*.ecr.dkr
  - Amazon S3 网关端点 – com.amazonaws.*region-code*.s3

  有关其他常用服务和端点的信息，请参阅 私有集群要求。
- 托管节点组无法部署在 AWS Outposts 上或者 AWS Wavelength 或 AWS 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 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 会返回错误代码来帮助您诊断问题。有关更多信息，请参阅 托管节点组错误代码。
- 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。
- 使用 AZRebalance 终止节点或减少所需节点数量时，未遵守容器组（pod）中断预算。这些操作试图驱逐节点上的 Pods。但如果超过 15 分钟，则无论节点上的所有 Pods 是否都被终止，该节点都会终止。要将时间延长至节点终止，请向自动扩缩组添加生命周期挂钩。有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的添加生命周期挂钩。
- 为了在收到 Spot 中断通知或容量再平衡通知后正确运行耗尽进程，必须将 CapacityRebalance 设置为 true。
- 更新托管节点组遵循您为 Pods 设置的 Pod 中断预算。有关更多信息，请参阅 托管节点更新行为。
- 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AWS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 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。

按需型

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 Capacity Type（容量类型），则会使用按需型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 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：

- 为确保在最佳 Spot 容量池中预置 Spot 节点，将分配策略设置为以下策略之一：
  - `price-capacity-optimized`（PCO）：在 Kubernetes 版本 `1.28` 或更高版本的集群中创建新节点组时，将分配策略设置为 `price-capacity-optimized`。但是，对于在 Amazon EKS 托管节点组开始支持 PCO 之前使用 `capacity-optimized` 创建的节点组，分配策略不会更改。
  - `capacity-optimized`（CO）：在 Kubernetes 版本 `1.27` 或更低版本的集群中创建新节点组时，将分配策略设置为 `capacity-optimized`。

  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。
  - 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 会尽最大努力耗尽节点。因此，无法保证 Amazon EKS 会等待替换节点加入集群后再耗尽现有节点。
  - 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。
- 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，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 使用启动模板自定义托管节点。

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 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 AWS Outposts、AWS Wavelength 或 AWS Local Zones 的 AWS 区域 创建托管节点。您可以在启用了 AWS Outposts、 AWS Wavelength 或 AWS Local Zones 的 AWS 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 节点启动自行管理的 Windows 节点 和 启动自行管理的 Bottlerocket 节点。您还可以在 Outpost 上创建自行管理的 Amazon Linux 节点组。有关更多信息，请参阅 在 Outpost 上启动自行管理的 Amazon Linux 节点。
> - 如果您没有为包含在 Amazon EKS 优化版 Linux 或 Bottlerocket 中的 `bootstrap.sh` 文件指定 AMI ID，则托管节点组会对 `maxPods` 的值强制实施最大数量。对于 vCPU 少于 30 个的实例，最大数量为 `110`。对于 vCPU 大于 30 个的实例，最大数量将跳至 `250`。这些数字基于Kubernetes可扩展性阈值以及内部 Amazon EKS 可扩展性团队测试的推荐设置。有关更多信息，请参阅博客文章 Amazon VPC CNI 插件提高每个节点的容器组限制。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 创建 Amazon EKS 集群。

- 供节点使用的现有 IAM 角色。要创建该文件，请参阅 Amazon EKS 节点 IAM 角色。如果此角色没有 VPC CNI 的任一策略，则需要为 VPC CNI Pod 使用随后的单独角色。
- （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 策略。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（IRSA）。
- 熟悉 选择 Amazon EC2 实例类型 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。
- 要添加 Windows 托管节点组，必须先启用对集群的 Windows 支持。有关更多信息，请参阅 为 Amazon EKS 集群启用 Windows 支持。

您可以使用 `eksctl` 或 AWS 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 `0.183.0` 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` 的说明，请参阅 `eksctl` 文档中的 Installation。

1. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM 策略附加到您的 Amazon EKS 节点 IAM 角色，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（IRSA）。
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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。

> **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的 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。

将 *`ami-family`* 替换为允许的关键字。有关更多信息，请参阅 `eksctl` 文档中的 Setting the node AMI Family（设置节点 AMI 系列）。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。

> **Note**
>
> 对于 Windows，此命令不启用 SSH。反之，它会将 Amazon EC2 密钥对与实例关联，并允许您 RDP 到实例中。

如果还没有 Amazon EC2 密钥对，可以在 AWS Management Console 中创建一个。有关 Linux 信息，请参阅 Amazon EC2 用户指南中的 Amazon EC2 密钥对和 Linux 实例。有关 Windows 信息，请参阅 Amazon EC2 用户指南中的 Amazon EC2 密钥对和 Windows 实例。

如果满足以下条件，我们建议阻止 Pod 访问 IMDS：

- 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
- 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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-ami-family ami-family \
  --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、容器组（pod）的自定义网络 和 私有集群要求，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 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 中指定的要求。

如果满足以下条件，我们建议阻止 Pod 访问 IMDS：

- 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
- 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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 *example values*，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、容器组（pod）的自定义网络、测试从 Docker 到 containerd 的迁移 和 私有集群要求，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 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或者想要增加分配到实例的 IP 地址数量）。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。

b. 使用以下命令部署节点组。

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

AWS Management Console

要使用 AWS 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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。
   - 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。

     > **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。

   - 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、容器组（pod）的自定义网络、测试从 Docker 到 `containerd` 的迁移 和 私有集群要求。

     启动模板必须满足 使用启动模板自定义托管节点 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，请将必要的权限直接分配到需要访问 AWS 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 AWS 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`**。有关更多信息，请参阅 托管节点组上的节点污点。
- 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。

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：选择 AMI 类型。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 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 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。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。

     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`、AWS CLI、AWS 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。
   - 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 以为您执行此操作。

   - 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。

- 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。这些节点在更新期间将不可用。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。
  - 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。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 Autoscaling 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 AWS CloudFormation 模板在 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 AWS 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 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。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。

> **Note**
>
> 对于 Windows，此命令不启用 SSH。反之，它会将 Amazon EC2 密钥对与实例关联，并允许您 RDP 到实例中。

- 对于 SSH 密钥对（可选），请选择要使用的 Amazon EC2 SSH 密钥。有关 Linux 信息，请参阅 Amazon EC2 用户指南中的 Amazon EC2 密钥对和 Linux 实例。有关 Windows 信息，请参阅 Amazon EC2 用户指南中的 Amazon EC2 密钥对和 Windows 实例。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。

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 优化加速型 AMI，则必须在集群上将适用于 Kubernetes 的 NVIDIA 设备插件用作 DaemonSet。将 *vX.X.X* 替换为您需要的 NVIDIA/k8s-device-plugin 版本，然后运行以下命令。

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

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 附加组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 主体是唯一可以使用 `kubectl` 或 AWS Management Console 调用 Kubernetes API 服务器的主体。如果您希望其他 IAM 主体拥有访问您的集群的权限，您需要添加它们。有关更多信息，请参阅授予对 Kubernetes API 的访问权限 和所需的权限。
- 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：
  - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。

- 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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。

- Autoscaling – 配置 Kubernetes Cluster Autoscaler 以自动调整节点组中的节点数。
- 将示例应用程序部署到您的集群。
- 集群管理 – 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 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
  - Amazon EKS 优化版 Bottlerocket AMI
  - Amazon ECS 优化版 Windows AMI 版本
- 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 AWS 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 AWS 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。
- 您已为 Windows 节点启用了 IP 前缀委派，并希望节点组中的新 AWS Nitro System 实例支持数量显著增加的 Pods。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量。

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。

更新节点组版本

您可以使用 `eksctl` 或 AWS Management Console 更新节点组。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 \
  --region=region-code
```

> **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 使用启动模板自定义托管节点 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 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 `1.29`，则您可以使用以下命令将当前运行 Kubernetes `1.28` 的节点升级到版本 `1.29`。

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.29
```

AWS Management Console

使用 AWS 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 发行版本）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）对话框中，激活或停用以下选项：

   - 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。
   - Change launch template version（更改启动模板版本）：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 Launch template version（启动模板版本）。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：

- 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。

6. 选择更新。

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）在编辑节点组页面上，执行以下操作：

   a. 编辑 Node group scaling configuration（节点组扩展配置）。

      - 所需大小 – 指定托管节点组应当维持的当前节点数量。
      - 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。

   b. （可选）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。

   c. （可选）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 **`NoSchedule`**、**`NoExecute`** 或 **`PreferNoSchedule`** 效果。有关更多信息，请参阅 托管节点组上的节点污点。

   d. （可选）向节点组资源添加或删除 Tags（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。

   e. （可选）编辑节点组更新配置。选择数字或百分比。

      - 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。

f. 编辑完成后，选择保存更改。

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. 它为与您的节点组关联的自动扩缩组创建新的 Amazon EC2 启动模板版本。新的启动模板版本使用目标 AMI 或自定义启动模板版本进行更新。
2. 它对自动扩缩组进行更新以使用最新的启动模板版本。
3. 它使用节点组的 `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. 它增加自动扩缩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：
   - 部署自动扩缩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（或者任何高于 10 的值），则该过程将启动 20 个新节点。
2. 扩展自动扩缩组后，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. 它将节点标记为不可调度以避免调度新的 Pods。它还使用 node.kubernetes.io/exclude-from-external-load-balancers=true 标记节点，以便在终止节点之前从负载均衡器中移除节点。

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。

账户的 EC2 实例限制

可能需要使用服务限额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。有关更多信息，请参阅 指定 AMI。

使节点不正常或未就绪的任何更改

节点磁盘压力、内存压力和类似情况可能导致节点无法进入 Ready 状态。

升级阶段

升级阶段有以下步骤：

1. 它随机选择一个需要更新的节点，最多为该节点组配置的最大不可用性。
2. 它耗尽节点的 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. 在每个 Pod 被驱逐后，它封锁节点并等待 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. 它向封锁节点的自动扩缩组发送终止请求。
5. 它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。

下面是导致这一阶段 `PodEvictionFailure` 出现错误的已知原因：

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。有关更多信息，请参阅 Kubernetes 文档中的污点和容忍度。

可以使用AWS 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。

- 有关使用 AWS Management Console 创建带有污点的节点组的信息，请参见 创建托管节点组。

- 以下是使用 AWS CLI 创建带有污点的节点组的示例：

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

有关更多信息和用法示例，请参阅 Kubernetes 参考文档中的污点。

Note

- 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
- 直接使用 Kubernetes 或 AWS Management Console 时，污点效果必须为 **NoSchedule**、**PreferNoSchedule** 或 **NoExecute**。但是，使用 AWS CLI 或 API 时，污点效果必须是 **NO_SCHEDULE**、**PREFER_NO_SCHEDULE** 或 **NO_EXECUTE**。
- 每个节点组最多允许 50 个污点。

- 如果从节点中手动删除使用托管节点组创建的污点，则 Amazon EKS 不会将污点重新添加回该节点。即使在托管节点组配置中指定了污点，也是如此。

您可以使用 `aws eks update-nodegroup-config` AWS CLI 命令为托管节点组添加、删除或替换污点。

使用启动模板自定义托管节点

要获得最高级别的自定义，您可以使用自己的启动模板部署托管节点。使用启动模板可以实现以下功能：

- 在部署节点时提供引导参数，例如额外的 `kubelet` 参数。
- 从与分配给节点的 IP 地址不同的 CIDR 块为 Pods 分配 IP 地址。
- 将您自己的自定义 AMI 部署到节点。
- 将您自己的自定义 CNI 部署到节点。

如果您在首次创建托管节点组时提供自己的启动模板，则以后也将具有更大的灵活性。只要使用自己的启动模板部署托管节点组，您就可以使用同一启动模板的不同版本对其进行迭代更新。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。

托管节点组始终部署用于 Amazon EC2 Auto Scaling 组的启动模板。当您不提供启动模板时，Amazon EKS API 会在您的账户中自动创建一个具有默认值的模板。但是，我们建议不要修改自动生成的启动模板。而且，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的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 AWS Management Console、AWS CLI 或AWS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。不要在两个位置都指定一个设置。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，它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 AWS 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
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 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）
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 AWS 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。 如果未在启动模板中指定 Application and OS Images (Amazon Machine Image)（应用程序和操作系统映像（Amazon 机器映像）） ，则可以在节点组配置中选择 AMI。	Launch template contents（启动模板内容）下的 Application and OS Images (Amazon Machine Image)（应用程序和操作系统映像（Amazon 机器映像））– 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 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，或将其部署到没有出站网络访问权

Amazon EKS 节点组配置 – 已禁止	启动模板
	限的私有集群。有关更多信息，请参阅以下主题： • 提高 Amazon EC2 节点的可用 IP 地址数量 • 容器组（pod）的自定义网络 • 私有集群要求 • 指定 AMI
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。

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。

- 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《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。
- 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。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用户数据部分的更多信息。

Amazon Linux 2 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==--
```

Amazon Linux 2023 user data

Amazon Linux 2023（AL2023）引入了使用 YAML 配置架构的新节点初始化流程 nodeadm。如果您使用的是自行管理的节点组或带有启动模板的 AMI，则在创建新节点组时，现在需要明确提供其他集群元数据。以下是最低必需参数的示例，其中 apiServerEndpoint、certificateAuthority 和服务 cidr 是必需的：

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

通常，您将在用户数据中设置此配置，无论是按原样设置还是嵌入 MIME 多部分文档中：

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

在 AL2 中，来自这些参数的元数据是从 Amazon EKS DescribeCluster API 调用中发现的。使用 AL2023，这种行为发生了变化，因为在大型节点纵向扩展期间，额外的 API 调用有节流风险。如果您使用的是没有启动模板的托管节点组或者正在使用 Karpenter，则此更改不会影响您。

有关 certificateAuthority 和服务 cidr 的更多信息，请参阅《Amazon EKS API 参考》中的 DescribeCluster。

Bottlerocket user data

Bottlerocket 采用 TOML 格式构建用户数据。您可以提供要与 Amazon EKS 提供的用户数据合并的用户数据。例如，您可以提供额外的 kubelet 设置。

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

有关支持设置的更多信息，请参阅 Bottlerocket 文档。您可以在用户数据中配置节点标签和污点。但是，我们建议在节点组中对其进行配置。在您这样做时，Amazon EKS 应用这些配置。

合并用户数据时，不保留格式，但内容保持不变。您在用户数据中提供的配置将覆盖 Amazon EKS 配置的任何设置。所以，如果设置 settings.kubernetes.max-pods 或 settings.kubernetes.cluster-dns-ip，用户数据中的这些值将应用到节点。

Amazon EKS 不支持所有有效的 TOM。以下是已知不受支持的格式的列表：

- 引用的键中的引号：'quoted "value"' = "value"
- 值中的转义引号：str = "I'm a string. \"You can quote me\""
- 混合浮点数和整数：numbers = [0.1, 0.2, 0.5, 1, 2, 5]
- 数组中的混合类型：contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]
- 带引用键的括号标题：[foo."bar.baz"]

Windows user data

Windows 用户数据使用 PowerShell 命令。创建托管节点组时，您的自定义用户数据与 Amazon EKS 托管用户数据结合使用。您的 PowerShell 命令排在首位，然后是托管用户数据命令，所有这些命令都在一个 **<powershell></powershell>** 标签内。

> **Note**
>
> 如果在启动模板中未指定 AMI ID，请勿在用户数据中使用 Windows Amazon EKS 引导脚本来配置 Amazon EKS。

用户数据示例如下所示。

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

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 Linux/Bottlerocket 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。例如，引导允许使用额外的 `kubelet` 参数。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`* 的优化版 AMI ID，您可以使用以下各部分中的表格：
  - 检索 Amazon EKS 优化版 Amazon Linux AMI ID
  - 检索 Amazon EKS 优化版 Bottlerocket AMI ID
  - 检索 Amazon EKS 优化版 Windows AMI ID
- 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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。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。

```
---
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 \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

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*）。
- 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。

```
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 \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

提供用户数据以将实际参数传递给随 Amazon EKS 优化版 Windows AMI 一起提供的 **Start-EKSBootstrap.ps1** 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 `Start-EKSBootstrap.ps1` 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。

如果您想指定自定义 Windows AMI ID，请记住以下注意事项：

- 您必须使用启动模板并在用户数据部分提供所需的引导命令。要检索所需的 Windows ID，您可以使用 Amazon EKS 优化版 Windows AMI 中的表。
- 有一些限制和条件。例如，您必须将 `eks:kube-proxy-windows` 添加到您的 AWS IAM 身份验证器配置映射中。有关更多信息，请参阅 指定 AMI ID 时的限制和条件。

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。`-APIServerEndpoint`、`-Base64ClusterCA` 和 `-DNSClusterIP` 参数是可选的。但是，定义它们可使 `Start-EKSBootstrap.ps1` 脚本避免进行 `describeCluster` 调用。

- 唯一必需的参数是集群名称（*my-cluster*）。
- 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
```

- 有关更多参数，请参阅 引导脚本配置参数。

> **Note**
>
> 如果您使用自定义服务 CIDR，则需要使用 -ServiceCIDR 参数进行指定。否则，集群 Pods 中的 DNS 解析将失败。

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

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《Amazon EC2 用户指南》中的亚马逊机器映像（AMI）。Amazon EKS AMI 构建规范包含用于构建基于 Amazon Linux 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。

指定 AMI ID 时的限制和条件

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`
- 如果您指定 AMI ID，则会异步应用 API 中的任何 `taints` 设置。要在节点加入集群之前应用污点，必须使用 `--register-with-taints` 命令行标志将污点传递到用户数据中的 `kubelet`。有关更多信息，请参阅 Kubernetes文档中的 kubelet。
- 为 Windows 托管节点组指定自定义 AMI ID 时，请将 `eks:kube-proxy-windows` 添加到您的 AWS IAM 身份验证器配置映射中。需要执行此操作 DNS 才能正常运行。

  1. 打开 AWS IAM 身份验证器配置映射进行编辑。

     ```
     kubectl edit -n kube-system cm aws-auth
     ```

  2. 将此条目添加到每个与 Windows 节点关联的 `rolearn` 下的 `groups` 列表中。您的配置映射应类似于 aws-auth-cm-windows.yaml。

     ```
     - eks:kube-proxy-windows
     ```

  3. 保存文件并退出文本编辑器。

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。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 中移除。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。如果您的集群的平台版本至少为 管理访问条目 先决条件部分中列出的最低版本，要为仅用于自行管理的节点组的角色添加映射，请参阅 创建访问条目。如果您的平台版本早于访问条目所需的最低版本，则可以将该条目重新添加到 aws-auth ConfigMap。要了解更多信息，请在您的终端中输入 eksctl create iamidentitymapping --help。

您可以使用 eksctl 或 AWS 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 \
  --name my-mng \
  --region region-code
```

有关更多选项，请参阅 eksctl 文档中的删除和清空节点组。

AWS Management Console

使用 AWS Management Console 删除托管的节点组

1. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
2. 在集群页面上，请选择包含要删除的节点组的集群。
3. 在所选集群页面上，请选择计算选项卡。
4. 在 Node groups（节点组）部分中，选择要删除的节点组。然后选择删除。
5. 在删除节点组确认对话框中，请输入节点组的名称。然后选择删除。

AWS CLI

使用 AWS CLI 删除托管的节点组

1. 输入以下 命令。将每个 *example value* 替换为您自己的值。

```
aws eks delete-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-mng \
  --region region-code
```

2. 使用键盘上的箭头键滚动浏览响应输出。完成后按 **q** 键。

有关更多选项，请参阅《AWS CLI 命令参考》中的 delete-nodegroup 命令。

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 AWS 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。您需要根据 Amazon EC2 价格收取费用。有关更多信息，请参阅 Amazon EC2 定价。

一个集群可以包含多个节点组。每个节点组都包含在 Amazon EC2 Auto Scaling 组中部署的一个或多个节点。组内节点的实例类型可能会有所不同，例如将基于属性的实例类型选择与 Karpenter 一起使用时。节点组中的所有实例都必须使用 Amazon EKS 节点 IAM 角色。

Amazon EKS 提供一个专用亚马逊机器映像（AMI），称为 Amazon EKS 优化版 AMI。AMI 配置为与 Amazon EKS 配合使用。其组件包括 containerd、kubelet 和 AWS 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。

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。

键	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
- 启动自行管理的 Bottlerocket 节点
- 启动自行管理的 Windows 节点
- 启动自行管理的 Ubuntu 节点
- 自行管理节点的更新

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 AWS Management Console 启动自行管理的 Amazon Linux 节点。如果您需要在 AWS Outposts 上启动节点，请参阅 在 Outpost 上启动自行管理的 Amazon Linux 节点。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 创建 Amazon EKS 集群。如果您在启用了 AWS Outposts、AWS Wavelength 或 AWS 本地区域的AWS 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。
- 供节点使用的现有 IAM 角色。要创建该文件，请参阅 Amazon EKS 节点 IAM 角色。如果此角色没有 VPC CNI 的任一策略，则需要为 VPC CNI Pod 使用随后的单独角色。
- （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 策略。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（IRSA）。
- 熟悉 选择 Amazon EC2 实例类型 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

> **Note**
>
> eksctl 目前不支持 Amazon Linux 2023。

先决条件

您的设备或 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。

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 策略附加到您的 Amazon EKS 节点 IAM 角色，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（IRSA）。
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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。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。

   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 AWS Management Console 中创建一个。有关更多信息，请参阅《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 AWS 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。
- 可以将 IPv4 地址分配给来自与实例不同的 CIDR 块的 Pods，请参阅 容器组（pod）的自定义网络。
- 可以将 IPv6 地址分配给 Pods 和服务，请参阅 集群、Pods 和 services 的 IPv6 地址。
- 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 测试从 Docker 到 containerd 的迁移。
- 没有出站互联网访问权限，请参阅 私有集群要求。

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。

示例输出如下。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）部署示例应用程序以测试集群和 Linux 节点。

4. 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
   - 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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。

AWS Management Console

步骤 1：要使用 AWS Management Console 启动自行管理 Linux 节点

1. 下载最新版本的 AWS CloudFormation 模板。

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/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 AWS 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）。
6. 选择您下载的 amazon-eks-nodegroup.yaml 文件。
7. 选择下一步。
8. 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 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - 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 时生成的 AWS 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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。
- 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。
- NodeImageIdSSMParam：使用用于某个变量 Kubernetes 版本最近的 Amazon EKS 优化版 AMI 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.XX* 替换为不同的支持版本。我们建议您指定与您的集群相同的 Kubernetes 版本。

  您也可以使用其他 AMI 类型替换 *amazon-linux-2*。有关更多信息，请参阅 检索 Amazon EKS 优化版 Amazon Linux AMI ID。

  > **Note**
  >
  > Amazon EKS 节点 AMI 基于 Amazon Linux。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 AWS 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）。
- NodeVolumeType：指定您的节点的根卷类型。
- 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 AWS Management Console 中创建一个。有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。

  > **Note**
  >
  > 如果此处不提供密钥对，AWS 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。
  - 可以将 `IPv4` 地址分配给来自与实例不同的 CIDR 块的 Pods，请参阅 容器组（pod）的自定义网络。
  - 可以将 `IPv6` 地址分配给 Pods 和服务，请参阅 集群、Pods 和 services 的 IPv6 地址。
  - 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 测试从 Docker 到 `containerd` 的迁移。
  - 没有出站互联网访问权限，请参阅 私有集群要求。
- 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 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 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 AWS 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 AWS CloudFormation VPC 模板部署的，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 AWS 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 中的注意事项和额外步骤。
> - 如果您选择 AWS 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。

9. 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）。
10. 选择 I acknowledge that AWS 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）。
11. 完成创建堆栈后，在控制台中选中它，然后选择 Outputs（输出）。
12. 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. 检查您是否已经应用拥有 `aws-auth ConfigMap`。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果您看到的是 `aws-auth ConfigMap`，则请根据需要对其进行更新。

a. 打开 ConfigMap 文件进行编辑。

```
kubectl edit -n kube-system configmap/aws-auth
```

b. 根据需要添加新的 mapRoles 条目。将 rolearn 值设置为您在上一个步骤中记录的 NodeInstanceRole 值。

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. 保存文件并退出文本编辑器。

3. 如果您收到错误提示 "Error from server (NotFound): configmaps "aws-auth" not found，则请使用库存 ConfigMap。

a. 下载配置映射。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm.yaml
```

b. 在 aws-auth-cm.yaml 文件中，将 rolearn 值设置为您在上一个步骤中记录的 NodeInstanceRole 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
```

4. 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。

> **Note**
>
> 如果您收到任何授权或资源类型错误，请参阅故障排除主题中的 未经授权或访问被拒绝 (kubectl)。

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。

5. （仅限 GPU 节点）如果选择 GPU 实例类型和 Amazon EKS 优化加速型 AMI，则必须在集群上将 适用于 Kubernetes 的 NVIDIA 设备插件 用作 DaemonSet。将 *vX.X.X* 替换为您需要的 NVIDIA/k8s-device-plugin 版本，然后运行以下命令。

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

第 3 步：其他操作

1. （可选）部署示例应用程序以测试集群和 Linux 节点。
2. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM 策略（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）附加到 the section called “节点 IAM 角色”，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（IRSA）。
3. 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：
   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
   - 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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。

ML 容量块

> **Important**
>
> - 容量块仅适用于某些 Amazon EC2 实例类型和 AWS 区域。有关兼容性信息，请参阅《Amazon EC2 用户指南（适用于 Linux 实例）》中的使用容量块。
> - 容量块目前不支持 Amazon EKS 托管节点组或 Karpenter。

机器学习（ML）容量块允许您在未来某个日期预留 GPU 实例，从而支持您的短期 ML 工作负载。在容量块内运行的实例会自动在 Amazon EC2 UltraClusters 中紧密放置，因此无需使用集群置放群组。有关更多信息，请参阅《Amazon EC2 用户指南（适用于 Linux 实例）》中的适用于 ML 的容量块。

您可以将容量块与 Amazon EKS 配合使用来预置和扩展自己的自行管理节点。以下步骤给出了一般的示例概述。

1. 在 AWS Management Console 中创建启动模板。有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的将容量块用于机器学习工作负载。

   请务必包括实例类型和 Amazon 系统映像（AMI）的配置。
2. 使用容量预留 ID 将容量块链接到启动模板。

   以下是针对容量块创建启动模板的示例 AWS CloudFormation 模板：

```
NodeLaunchTemplate:
    Type: "AWS::EC2::LaunchTemplate"
    Properties:
      LaunchTemplateData:
        InstanceMarketOptions:
          MarketType: "capacity-block"
        CapacityReservationSpecification:
          CapacityReservationTarget:
            CapacityReservationId: "cr-02168da1478b509e0"
        IamInstanceProfile:
          Arn: iam-instance-profile-arn
        ImageId: image-id
        InstanceType: p5.48xlarge
        KeyName: key-name
        SecurityGroupIds:
        - sg-05b1d815d1EXAMPLE
```

```
        UserData: user-data
```

由于容量块是区域性的，因此必须将子网传递到进行预留的可用区。

3. 如果您在容量预留变为活动状态之前创建自行管理节点组，请将所需容量设置为 0。创建节点组时，请确保仅为预留容量的可用区指定相应的子网。

   以下是可供使用的 CloudFormation 模板示例。此示例获取上一个示例中所显示 AWS::Amazon EC2::LaunchTemplate 资源的 LaunchTemplateId 和 Version。它还会获取 DesiredCapacity、MaxSize、MinSize 和 VPCZoneIdentifier 的值，这些内容在同一模板的其他位置声明。

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
        PropagateAtLaunch: true
        Value: owned
```

4. 成功创建节点组后，请务必记录已创建节点组的 NodeInstanceRole。您需要此信息以确保在扩展节点组时，新节点会加入集群，并且 Kubernetes 能够识别节点。有关更多信息，请参阅 启动自行管理的 Amazon Linux 节点 中的 AWS Management Console 说明。

5. 我们建议您为 Auto Scaling 组创建与容量块预留时间保持一致的计划扩展策略。有关更多信息，请参阅 Amazon EC2 Auto Scaling 用户指南中的 Amazon EC2 Auto Scaling 的计划扩展。

   您可以在容量块结束时间前 30 分钟使用预留的所有实例。那时仍在运行的实例将开始终止。为了留出足够的时间来正常地耗尽节点，我们建议您在容量块预留结束时间前 30 分钟计划扩展至零。

   如果您想改为在容量预留变为 Active 时手动纵向扩展，则需要在容量块预留开始时更新 Auto Scaling 组的所需容量。然后，您还需要在容量块预留结束前 30 多分钟手动缩减。

6. 节点组现在已准备好计划工作负载和 Pods。
7. 为了正常地耗尽您的 Pods，我们建议您设置 AWS 节点终止处理程序。该处理程序将能够使用 EventBridge 监视来自 Amazon EC2 Auto Scaling 的“ASG 横向缩减”生命周期事件，并允许 Kubernetes 控制面板在实例不可用之前执行所需的操作。否则，您的 Pods 和 Kubernetes 对象将停留在待处理状态。有关更多信息，请参阅 GitHub 上的 AWS 节点终止处理程序。

   如果您没有设置节点终止处理程序，我们建议您在到达 30 分钟时段之前开始手动耗尽 Pods，这样就有充足的时间正常地耗尽这些节点。

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。

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 AWS 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。
> - 您可以在 AWS Outpost 上的 Amazon EKS 扩展集群中启动 Bottlerocket 节点，但您无法在 AWS Outpost 上的本地集群中启动它们。有关更多信息，请参阅 AWS Outposts 上的 Amazon EKS。
> - 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。
> - Bottlerocket 与 AWS CloudFormation 兼容。但是，没有可以复制用于为 Amazon EKS 部署 Bottlerocket 节点的官方 CloudFormation 模板。

- Bottlerocket 映像不附带 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 `0.183.0` 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` 的说明，请参阅 `eksctl` 文档中的 Installation。

> **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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。将 *`ng-bottlerocket`* 替换为您的节点组名称。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。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 AWS Management Console 中创建一个。有关更多信息，请参阅《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 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 使用启动模板自定义托管节点。

> **Important**
>
> 要将节点组部署到 AWS Outposts、AWS Wavelength 或 AWS 本地区域子网，创建集群时不要传入 AWS Outposts、AWS Wavelength 或 AWS 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 AWS 区域。

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.30'

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
```

示例输出如下。

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。
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）部署示例应用程序来测试您的 Bottlerocket 节点。
6. 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
   - 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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。
> - 您可以在 AWS Outpost 上的 Amazon EKS 扩展集群中启动 Windows 节点，但您无法在 AWS Outpost 上的本地集群中启动它们。有关更多信息，请参阅 AWS Outposts 上的 Amazon EKS。

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。

您可以通过 `eksctl` 或 AWS 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 `0.183.0`。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` 的说明，请参阅 `eksctl` 文档中的 Installation。

> **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 策略（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）附加到 the section called “节点 IAM 角色”，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（IRSA）。

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 节点组，则建议您遵循 开始使用 Amazon EKS – eksctl 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 *region-code* 替换为集群所在的 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。将 *ng-windows* 替换为您的节点组名称。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。对于 Kubernetes 1.24 版本或更高版本，您可以将 *2019* 替换为 2022 以使用 Windows Server 2022。将剩余的 *example values* 替换为您自己的值。

   > **Important**
   >
   > 要将节点组部署到 AWS Outposts、AWS Wavelength 或 AWS 本地区域子网，在创建集群时不要传递 AWS Outposts、Wavelength 或本地区域子网。使用配置文件创建节点组，指定 AWS 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。

   ```
   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。
   > - 要查看 eksctl 命令可用选项，请输入以下命令。

```
eksctl command -help
```

示例输出如下。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）部署示例应用程序以测试集群和 Windows 节点。
4. 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：
   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
   - 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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。

AWS 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 和 Amazon EKS 安全组要求和注意事项。开始使用 Amazon EKS 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 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。如果您在启用了 AWS Outposts、AWS Wavelength 或 AWS Local Zones 的 AWS 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 AWS 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 AWS CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
3. 选择创建堆栈。

4. 对于 Specify template（指定模板），选择 Amazon S3 URL。
5. 复制以下 URL 并将其粘贴到 Amazon S3 URL 中。

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. 选择 Next（下一步）两次。
7. 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 AWS 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 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 时生成的 AWS 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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。
   - 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。

  > **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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。

- 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 AWS 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 AWS Management Console 中创建一个。有关更多信息，请参阅《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，AWS 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。
- 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 的 ID。
- NodeSecurityGroups：选择在您创建 VPC 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 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 AWS 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 AWS CloudFormation VPC 模板部署的，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 AWS 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 中的注意事项和额外步骤。
> - 如果您选择 AWS 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。

8. 确认堆栈可创建 IAM 资源，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 Windows 节点时需要此值。

步骤 2：使节点能够加入集群

1. 检查您是否已经应用拥有 `aws-auth ConfigMap`。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果您看到的是 `aws-auth ConfigMap`，则请根据需要对其进行更新。

   a. 打开 `ConfigMap` 文件进行编辑。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 根据需要添加新的 `mapRoles` 条目。将 `rolearn` 值设置为您在前面的步骤中记录的 NodeInstanceRole 值。

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

c. 保存文件并退出文本编辑器。

3. 如果您收到错误提示 "Error from server (NotFound): configmaps "aws-auth" not found，则请使用库存 ConfigMap。

a. 下载配置映射。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
```

b. 在 aws-auth-cm-windows.yaml 文件中，将 rolearn 值设置为您在前面的步骤中记录的 NodeInstanceRole 值。您可以使用文本编辑器或者通过替换 *example values* 和运行以下命令来执行此操作：

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-
node-linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-
windows-instance-role|' aws-auth-cm-windows.yaml
```

> **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。

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
```

4. 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。

> **Note**
>
> 如果您收到任何授权或资源类型错误，请参阅故障排除主题中的 未经授权或访问被拒绝 (kubectl)。

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。

第 3 步：其他操作

1. （可选）部署示例应用程序以测试集群和 Windows 节点。
2. （可选）如果 AmazonEKS_CNI_Policy 托管 IAM 策略（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）附加到 the section called “节点 IAM 角色”，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（IRSA）。
3. 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：
   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
   - 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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 Ubuntu 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。

本主题描述如何启动向 Amazon EKS 集群注册的 Ubuntu on Amazon Elastic Kubernetes Service（EKS） 或 Ubuntu Pro on Amazon Elastic Kubernetes Service（EKS）的自动扩缩组。适用于 EKS 的 Ubuntu 和 Ubuntu Pro 基于官方的 Ubuntu Minimal LTS，包括与 AWS 共同开发并专门为 EKS 构建的自定义 AWS 内核。Ubuntu Pro 通过支持 EKS 延长支持期、内核 livepatch、FIPS 合规性以及运行无限 Pro 容器的能力，增加了额外的安全保障。

在这些节点加入集群后，您可以向其部署容器化应用程序。有关更多信息，请访问有关 AWS 上的 Ubuntu 的文档和 `eksctl` 文档中的自定义 AMI 支持。

> **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。
> - 您可以在 Ubuntu Outpost 上的 Amazon EKS 扩展集群中启动 AWS 节点，但您无法在 AWS Outpost 上的本地集群中启动它们。有关更多信息，请参阅 AWS Outposts 上的 Amazon EKS。
> - 您可以部署到采用 x86 或 Arm 处理器的 Amazon EC2 实例。但是，具有 Inferentia 芯片的实例可能需要先安装 Neuron SDK。

使用 **`eksctl`** 启动用于 EKS 的 Ubuntu 或用于 EKS 节点的 Ubuntu Pro

此过程需要 `eksctl` 版本 `0.183.0` 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` 的说明，请参阅 `eksctl` 文档中的 Installation。

> **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-ubuntu` 替换为您的节点组名称。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。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 AWS Management Console 中创建一个。有关更多信息，请参阅《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 `ubuntu.yaml` 文件。

   > **Important**
   >
   > 要将节点组部署到 AWS Outposts、AWS Wavelength 或 AWS 本地区域子网，创建集群时不要传入 AWS Outposts、AWS Wavelength 或 AWS 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 AWS 区域。

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.30'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
```

```
    desiredCapacity: 3
    amiFamily: Ubuntu22.04
    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
```

要创建 Ubuntu Pro 节点组，只需将 `amiFamily` 值更改为 `UbuntuPro2204`。

2. 使用以下命令部署您的节点。

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

示例输出如下。

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）部署示例应用程序来测试您的 Ubuntu 节点。
4. 如果满足以下条件，我们建议阻止 Pod 访问 IMDS：

   - 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
   - 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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，请参阅 更新托管节点组。

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

创建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 AWS 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

更新现有节点组的 AWS 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 AWS Management Console 迁移到新的节点组。

eksctl

使用 **`eksctl`** 将您的应用程序迁移到新的节点组

有关使用 eksctl 进行迁移的更多信息，请参阅 `eksctl` 文档中的非托管节点组。

此过程需要 `eksctl` 版本 `0.183.0` 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` 的说明，请参阅 `eksctl` 文档中的 Installation。

> **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
```

示例输出如下。

```
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 Pod 访问 IMDS：

- 您计划将 IAM 角色分配到所有 Kubernetes 服务账户，以便 Pods 只具有所需的最低权限。
- 集群中没有任何 Pods 需要出于其他原因（例如检索当前 AWS 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 1.30 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

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-old
```

AWS Management Console and AWS CLI

使用 AWS Management Console 和 AWS 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 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。

   a. 记下两个节点组的安全组 ID。这在 AWS CloudFormation 堆栈输出中显示为 NodeSecurityGroup 值。

   您可以使用以下 AWS 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 AWS 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 AWS 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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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。有关更多信息，请参阅 Kubernetes 文档中的污点和容忍度。

```
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 1.28）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=1.28
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
```

示例输出如下。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
```

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 *1.28*）的所有节点。

```
K8S_VERSION=1.28
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 AWS 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 AWS 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 AWS 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 AWS 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 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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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 AWS 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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。
16. 如果您的集群使用适用于 DNS 解析的 `kube-dns`（请参阅上一步），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 AWS 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 的信息，请参阅 更新托管节点组。

最新的默认 Amazon EKS 节点 AWS 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。

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
   ```

   示例输出如下。此集群使用 CoreDNS 解析 DNS，但您的集群可能会返回 kube-dns。您的输出可能看起来有所不同，具体取决于您使用的 kubectl 版本。

   ```
   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 AWS 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 AWS 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 AWS CloudFormation 模板。接下来，选择 Next（下一步）：

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/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。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 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 Worker 节点和 Pods 的 30 个 IP 地址。

> **Note**
>
> GitHub 上的 vpc_ip_resource_limit.go 中显示 Amazon VPC CNI plugin for Kubernetes 最新版本支持的实例类型。您可能需要更新 Amazon VPC CNI plugin for Kubernetes 版本以使用最新的受支持实例类型。有关更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。

> **Important**
>
> 并非所有实例类型在所有 AWS 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 `1.30` 使用最新 Amazon EKS 优化版 AMI。

```
/aws/service/eks/optimized-ami/1.30/amazon-linux-2/recommended/image_id
```

您可以将 *1.30* 替换为支持的 Kubernetes 版本。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。您也可以使用其他 AMI 类型替换 *amazon-linux-2*。有关更多信息，请参阅 检索 Amazon EKS 优化版 Amazon Linux AMI ID。

> **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 AWS 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 AWS 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 AWS 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 Amazon VPC CNI plugin for Kubernetes。您可能需要更新 Amazon VPC CNI plugin for Kubernetes 版本以使用最新的受支持实例类型。有关更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。

AWS Fargate

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

本主题讨论使用 Amazon EKS 在 AWS Fargate 上运行 Kubernetes Pods。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 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 配置文件运行。Fargate 配置文件被定义为 Amazon EKS 集群的一部分。Amazon EKS 使用由 AWS 构建的控制器（使用 Kubernetes 提供的上游可扩展模型）将 Kubernetes 与 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。

AWS Fargate 注意事项

以下是使用 Amazon EKS 上的 Fargate 时要考虑的一些事项。

- 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 和 Amazon EKS 上的应用程序负载均衡。
- 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。
- 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AWS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。
- 您可以使用 Vertical Pod Autoscaler 为 Fargate Pods 设置正确的初始 CPU 和内存大小，然后使用 Horizontal Pod Autoscaler 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 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 AWS 区域 或可用区。
- 您不能将 Fargate Pods 部署到 AWS Outposts、AWS Wavelength 或 AWS Local Zones。
- Amazon EKS 必须定期修补 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 操作系统修补。
- 适用于 Amazon EKS 的 Amazon VPC CNI 插件安装在 Fargate 节点上。您不可以将 备选的兼容 CNI 插件 与 Fargate 节点一起使用。
- 在 Fargate 上运行的 Pod 会自动挂载 Amazon EFS 文件系统。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。
- 您无法将 Amazon EBS 卷挂载到 Fargate Pods。
- 您可以在 Fargate 节点上运行 Amazon EBS CSI 控制器，但要在 Amazon EBS CSI 节点 DaemonSet 只能在 Amazon EC2 实例上运行。
- 在 Kubernetes Job 被标记为 `Completed` 或 `Failed` 后，Job 创建的 Pods 通常会继续存在。此行为允许您查看日志和结果，但在使用 Fargate 的情况下，如果您过后不清理 Job，则将会产生费用。

  要在 Job 完成或失败后自动删除相关的 Pods，您可以使用生存时间（TTL）控制器指定时间段。以下示例显示了在 Job 清单中指定 `.spec.ttlSecondsAfterFinished`。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

通过 Amazon EKS 开始使用 AWS Fargate

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

本主题描述如何通过 Amazon EKS 集群开始在 AWS 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。

先决条件

现有集群。如果您还没有 Amazon EKS 集群，请参阅 开始使用 Amazon EKS。

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的集群，可以跳至 创建 Fargate Pod 执行角色。

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 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 AWS 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 AWS 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 AWS 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 AWS 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 AWS API。Amazon EKS Pod 执行角色提供执行此操作的 IAM 权限。要创建 AWS Fargate Pod 执行角色，请参阅 Amazon EKS Pod 执行 IAM 角色。

> **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 AWS Fargate 配置文件。

> **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 AWS Management Console 创建 Fargate 配置文件。

eksctl

此过程需要 `eksctl` 版本 `0.183.0` 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` 的说明，请参阅 `eksctl` 文档中的 Installation。

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。

AWS Management Console

使用 AWS 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。
   c. 根据需要修改选定的子网。

      > **Note**
      >
      > 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。

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。

c. 选择下一步。

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。

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。

> **Important**
>
> 角色 ARN 不能包含 / 之外的路径。例如，如果您的角色名称为 development/apps/my-role，则需要在为该角色指定 ARN 时将其更改为 my-role。角色 ARN 的格式必须为 arn:aws:iam::*111122223333*:role/*role-name*。

```
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，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 以允许在 Fargate 上运行的 Pods 的入口对象。
- 您可以使用 Vertical Pod Autoscaler 为 Fargate Pods 设置正确的初始 CPU 和内存大小，然后使用 Horizontal Pod Autoscaler 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 AWS Distro for OpenTelemetry (ADOT) 收集器，用于监控应用程序。

AWS Fargate 配置文件

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

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。每个选择器都必须包含一个命名空间。选择器还可以包含标签。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。

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。

Pod 执行角色

当您的集群在 AWS Fargate 上创建 Pods 时，在 Fargate 基础设施上运行的 kubelet 必须代表您调用 AWS 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。

子网

将 Pods 启动到其中使用此配置文件的子网 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 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 Pods 使用，则 Fargate 会为 Pods 选取配置文件 A（beta-workload）。Pods 具有标签，显示容器组 Pods 上有配置文件 A（例如 eks.amazonaws.com/fargate-profile=beta-workload）。

如果要将现有 Fargate Pods 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 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 Pods。

创建 Fargate 配置文件

本主题介绍如何创建 Fargate 配置文件。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。在 Fargate 上运行的 Pods 仅在私有子网上受支持（对 AWS 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 必须有可用的私有子网。您可以使用 eksctl 或 AWS Management Console 创建配置文件。

此过程需要 eksctl 版本 0.183.0 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 的说明，请参阅 eksctl 文档中的 Installation。

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。

AWS Management Console

使用 AWS 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。
   c. 根据需要修改选定的子网。

      > **Note**
      >
      > 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。

   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。

   c. 选择下一步。

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、AWS Management Console 或 AWS 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
```

AWS Management Console

利用 AWS 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）。

AWS CLI

使用 AWS 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

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

本节介绍在 AWS Fargate 上运行 Kubernetes Pods 的一些唯一 Pod 配置的详细信息。

Pod CPU 和内存

借助 Kubernetes，您可以定义 Pod 中的请求、最小 vCPU 数量以及分配给每个容器中的内存资源。Pods 由 Kubernetes 进行调度，以确保计算资源中至少有每个 Pod 请求的资源可用。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 Pods 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 256MB。

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（0.25 vCPU 和 0.5GB 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 个 vCPU	2GB、3GB、4GB、5GB、6GB、7GB、8GB
2 个 vCPU	4GB 到 16GB 之间（以 1GB 为增量）
4 个 vCPU	8GB 到 30GB 之间（以 1GB 为增量）
8 个 vCPU	介于 16GB 到 60GB 之间（以 4GB 为增量）
16 个 vCPU	介于 32GB 到 120GB 之间（以 8GB 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 9GB 内存的任务可用，因此如果请求内容为 1 个 vCPU 和 8GB 内存，则会为其内存请求增加 256MB，并会为 Fargate 任务预置 2 个 vCPU 和 9GB 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。请将 *`default`* 替换为您的 Pod 命名空间，并将 *`pod-name`* 替换为您的 Pod 名称。

```
kubectl describe pod --namespace default pod-name
```

示例输出如下。

```
[...]
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 AWS Fargate 定价。

Fargate 存储

在 Fargate 上运行的 Pod 会自动挂载 Amazon EFS 文件系统。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。有关更多信息，请参阅 GitHub 上的 Amazon EFS CSI 驱动程序。

完成预置后，在 Fargate 上运行的每个 Pod 都会获得 20GiB 的默认临时存储空间。这种类型的存储将在 Pod 停止后被删除。在 Fargate 上启动的新 Pods 会默认启用临时存储卷加密。临时 Pod 存储将使用 AWS Fargate 托管式密钥和 AES-256 加密算法进行加密。

> **Note**
>
> 在 Fargate 上运行的 Amazon EKS Pods 的默认可用存储空间值小于 20GiB。这是因为部分空间会被 `kubelet` 以及 Pod 内部加载的其他 Kubernetes 模块使用。

您最高可以将临时存储总量增加到 175GiB。要使用 Kubernetes 配置大小，请指定一个 Pod 中每个容器的 `ephemeral-storage` 资源请求数。在 Kubernetes 调度 Pods 时，它会确保每个 Pod 的资源请

求总和小于 Fargate 任务的容量。有关更多信息，请参阅 Kubernetes 文档中的为 Pods 和容器资源管理。

Amazon EKS Fargate 预置的临时存储空间会高于请求的容量，以满足系统使用的需要。例如，假设您请求 100GiB 的空间，则系统会为 Fargate 任务预置 115GiB 的临时存储空间。

Fargate 操作系统修补

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

Amazon EKS 定期修补 AWS Fargate 节点的操作系统以确保它们的安全。作为修补过程的一部分，我们会回收节点来安装操作系统补丁。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 Amazon EventBridge 规则来在删除 Pods 之前处理失败的驱逐。
- 在 AWS 用户通知中创建通知配置。

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。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 可能会回收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。

要限制在回收 Pods 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 AWS 账户 和集群所属的 AWS 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。您还可以在 AWS 用户通知中创建通知配置。使用 AWS Management Console 创建通知时，在事件规则下，为AWS 服务 名称选择 Elastic Kubernetes Service (EKS)，为事件类型选择 EKS Fargate 容器组（pod）计划的终止。有关更多信息，请参阅《AWS 用户通知用户指南》中的AWS 用户通知入门。

Fargate 指标

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

您可以收集系统指标和 AWS Fargate 的 CloudWatch 使用量指标。

应用程序指标

对于在 Amazon EKS 和 AWS Fargate 上运行的应用程序，您可以使用 AWS 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 AWS 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。

AWS Fargate 用量指标与 AWS 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。

AWS Fargate 在 `AWS/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 AWS Fargate 发布的用量指标。

维度	描述
`Service`	包含该资源的 AWS 服务的名称。对于 AWS 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，AWS 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，AWS 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，AWS Fargate 不使用类维度。

创建 CloudWatch 警报以监控 Fargate 资源使用情况指标

AWS 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AWS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。

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 (AWS 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 AWS 服务。
3. 从 AWS services（亚马逊云科技服务）列表中，搜索并选择 AWS 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

> **Important**
>
> 具有 Amazon EKS 的 AWS Fargate 在 AWS GovCloud（美国东部）和 AWS GovCloud（美国西部）不可用。

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
- 不能超过 5300 个字符。

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 AWS for Fluent Bit 版本是 Fluent Bit 的一种上游合规发行版，由 AWS 托管。有关更多信息，请参阅 GitHub 上的 AWS for Fluent Bit。

日志路由器允许您使用广泛的AWS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 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 Pods 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。
- 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。

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。
   > - Parser 部分。
   >
   > 如果您提供任何其他部分，将被拒绝。

Fargate 日志路由器管理 Service 和 Input 部分。它包含以下 Input 部分，无法修改，并且您的 ConfigMap 也不需要该部分。但是，您可以从中获取见解，例如内存缓冲区限制和应用于日志的标签。

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

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 Pods。不会将更改应用于现有 Pods。

使用所需日志目标的示例创建 ConfigMap。

> **Note**
>
> 您也可以将 Amazon Kinesis Data Streams 用作您的日志目的地。如果您使用 Kinesis Data Streams，则请确保容器组（pod）执行角色已被授予 kinesis:PutRecords 权限。有关更多信息，请参阅 Fluent Bit：官方手册中的 Amazon Kinesis Data Streams 权限。

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 AWS 区域。[OUTPUT] 下的参数是必需的。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
```

```
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  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
```

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 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 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？

Firehose

为 Firehose 创建 **ConfigMap**

将日志发送到 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。

以下示例向您展示了如何使用 `kinesis_firehose` 插件将日志发送到 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 AWS 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 Firehose IAM 策略下载到您的计算机。您还可以在 GitHub 上查看策略。

```
curl -O 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 Pod 执行角色（有关更多信息，请参阅 创建 Fargate Pod 执行角色）。

```
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
1.23 和更高版本	eks.1

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

您可以选择性地使用下面的 `ConfigMap` 将 Fluent Bit 流程日志发送到 Amazon CloudWatch。将 Fluent Bit 进程日志发送到 CloudWatch 需要额外的日志摄取和存储成本。将 *`region-code`* 替换为集群所在的 AWS 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 AWS 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。

要停止发送 Fluent Bit 进程日志

将 Fluent Bit 进程日志发送到 CloudWatch 需要额外的日志摄取和存储成本。要排除现有 `ConfigMap` 设置中的进程日志，请执行以下步骤。

1. 在启用 Fargate 日志记录后，找到为 Amazon EKS 集群的 Fluent Bit 进程日志自动创建的 CloudWatch 日志组。它遵循格式 `{cluster_name}-fluent-bit-logs`。
2. 删除 CloudWatch 日志组中为每个 Pod's 的进程日志创建的现有 CloudWatch 日志流。
3. 编辑 `ConfigMap` 并设置 `flb_log_cw: "false"`。
4. 重启集群中的任何现有 Pods。

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 50MB 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 100MB 的内存。

故障排除

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 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 事件是短暂的，时间长短取决于设置。您也可以使用 **kubectl describe pod *pod-name*** 查看 Pod's 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 Amazon EC2 用户指南中的可用实例类型和 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。
- 对于基于 ARM 的实例，Amazon Linux 2023（AL2023）仅支持使用 Graviton2 或更高版本处理器的实例类型。AL2023 不支持 `A1` 实例。

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 为 Amazon EKS 集群启用 Windows 支持。

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 Amazon EKS 优化版 Arm Amazon Linux AMI。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。

> **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。

AWS 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。要为实例分配最大数量的 IP 地址，您必须在集群中安装 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 AWS Nitro System 实例类型或裸机实例类型。Windows 实例仅支持 IPv4。您的集群必须运行 1.10.1 版或更高版本的 Amazon VPC CNI 附加组件。有关使用 IPv6 的更多信息，请参阅集群、Pods 和 services 的 IPv6 地址。

您正在运行的 Amazon VPC CNI 附加组件的版本

适用于 Kubernetes 的最新版本的 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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。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。

您在其中创建节点的 AWS 区域

并非所有实例类型在所有 AWS 区域 中都可用。

您是否将安全组用于 Pods

如果您将安全组用于 Pods，则仅支持特定的实例类型。有关更多信息，请参阅 Pods 的安全组。

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。您可以使用这些步骤中返回的数字来启用诸如将 IP 地址分配给不同于实例子网的 Pods 和显著增加实例的 IP 地址数量等功能。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 https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/al2/runtime/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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件 中的更新过程。

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

   示例输出如下。

```
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 容器组（pod）的自定义网络。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`。
   - `--cni-prefix-delegation-enabled`：如果要为每个弹性网络接口分配数量显著增加的 IP 地址，请使用此选项。此功能需要在 Nitro System 上运行的 Amazon Linux 实例和 `1.9.0` 或更高版本的 Amazon VPC CNI 附加组件。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量。将此选项添加到具有相同示例值的上一个脚本中会生成 `110`。

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。

> **Note**
>
> 最大 Pods 计算器脚本根据 Kubernetes 可扩展性阈值和推荐设置将返回值限制为 `110`。如果您的实例类型超过 30 个 vCPU，则此限制将跳至 `250`，这是基于内部 Amazon EKS 可扩展性

团队测试得出的数字。有关更多信息，请参阅博客文章 Amazon VPC CNI 插件提高每个节点的容器组限制。

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。

主题

- Amazon EKS 结束了对 Dockershim 的支持
- Amazon EKS 优化版 Amazon Linux AMI
- Amazon EKS 优化版 Bottlerocket AMI
- Amazon EKS 优化版 Ubuntu Linux AMI
- Amazon EKS 优化版 Windows AMI

Amazon EKS 结束了对 **Dockershim** 的支持

Kubernetes 不再支持 `Dockershim`。Kubernetes 团队移除了 Kubernetes 版本 `1.24` 中的运行时。有关更多信息，请参阅 Kubernetes 博客上的 Kubernetes is Moving on From Dockershim: Commitments and Next Steps。

Amazon EKS 还将从 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。此主题涵盖了一些细节，有关更多信息请参见 迁移到 Amazon EKS 上的 containerd 详解。

您可以使用 `kubectl` 插件查看您的哪些 Kubernetes 工作负载挂载了 Docker 套接字卷。有关更多信息，请参阅 GitHub 上的 Detector for Docker Socket (DDS)。如果 Amazon EKS AMI 运行的 Kubernetes 版本早于 `1.24`，则使用 Docker 作为默认运行时。但是，这些 Amazon EKS AMI 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 测试从 Docker 到 `containerd` 的迁移。

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。如果需要更多时间在 `containerd` 上测试工作负载，请使用 `1.24` 版之前的支持版本。但是，当您想将官方 Amazon EKS AMI 升级到版本 `1.24` 或更高版本时，请确保验证您的工作负载可以在 `containerd` 上运行。

containerd 运行时提供更可靠的性能和安全性。containerd 是整个 Amazon EKS 标准化的运行时。Fargate 和 Bottlerocket 已经只能使用 containerd。containerd 有助于最大程度地减少 Dockershim 常见漏洞和风险（CVE）所需的 Amazon EKS AMI 发布次数。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 套接字的应用程序进行更改。例如，使用容器构建的容器映像将受到影响。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 设置的应用程序进行更改。例如，不再支持 HTTPS_PROXY 协议。您必须更新使用此协议的应用程序。有关更多信息，请参阅 Docker 文档中的 dockerd。
- 如果您使用 Amazon ECR 凭证助手提取映像，则必须切换到 kubelet 映像凭证提供程序。有关更多信息，请参阅 Kubernetes 文档中的配置 kubelet 映像凭证提供程序。
- 由于 Amazon EKS 1.24 不再支持 Docker，因此不再支持 Amazon EKS 引导脚本 以前支持的某些标志。在迁移到 Amazon EKS 1.24 或更高版本之前，您必须删除任何引用现在不支持的标志的内容：
  - --container-runtime dockerd（containerd 是唯一受支持的值）
  - --enable-docker-bridge
  - --docker-config-json
- 如果您已经为 Container Insights 进行了 Fluentd 配置，则必须先将 Fluentd 迁移到 Fluent Bit，然后才能更改为 containerd。Fluentd 解析器配置为仅解析 JSON 格式的日志消息。与 dockerd 不同的是，containerd 容器运行时系统的日志消息不是 JSON 格式的。如果您不迁移到 Fluent Bit，一些配置的 Fluentd's 解析器将在 Fluentd 容器内生成大量错误。有关迁移的更多信息，请参阅将 Fluent Bit 设置为 DaemonSet 以将日志发送到 CloudWatch Logs。
- 如果您使用自定义 AMI 并且要升级到 Amazon EKS 1.24，则必须确保为您的 Worker 节点启用 IP 转发。Docker 不需要此设置，但 containerd 需要。需要对 Pod 到 Pod、Pod 到外部或 Pod 到 apiserver 的网络连接进行故障排除。

  要在 Worker 节点上验证此设置，请运行以下任一命令：
  - sysctl net.ipv4.ip_forward
  - cat /proc/sys/net/ipv4/ip_forward

  如果输出为 0，则运行以下任一命令来激活 net.ipv4.ip_forward 内核变量：
  - sysctl -w net.ipv4.ip_forward=1
  - echo 1 > /proc/sys/net/ipv4/ip_forward

有关在 containerd 运行时系统中的 Amazon EKS AMI 上激活该设置的情况，请参阅 GitHub 上的 install-worker.sh。

Amazon EKS 优化版 Amazon Linux AMI

Amazon EKS 优化的 Amazon Linux AMI 基于 Amazon Linux 2（AL2）和 Amazon Linux 2023（AL2023）构建。配置作为 Amazon EKS 节点的基本映像。AMI 配置为与 Amazon EKS 搭配使用，它包含以下组件：

- kubelet
- AWS IAM 身份验证器
- Docker（Amazon EKS 版本 1.23 及更早版本）
- containerd

Note

- 您可以在 Amazon Linux 安全中心跟踪 AL2 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 和 Amazon EKS 优化版 Arm Amazon Linux AMI 中的信息。
- 对于 Kubernetes 版本 1.23，您可以使用可选引导标记测试从 Docker 迁移到 containerd。有关更多信息，请参阅 测试从 Docker 到 containerd 的迁移。
- 从 Kubernetes 版本 1.25 开始，您将无法再将 Amazon EC2 P2 实例与现成的 Amazon EKS 优化加速 Amazon Linux AMI 结合使用。对于 Kubernetes 版本 1.25 或更高版本，这些 AMI 将支持 NVIDIA 525 系列或更高版本的驱动程序，这些驱动程序与 P2 实例不兼容。但 NVIDIA 525 系列或更高版本的驱动程序与 P3、P4 和 P5 实例兼容，因此您可以通过 Kubernetes 版本 1.25 或更高版本的 AMI 使用这些实例。请首先将所有 P2 实例迁移到 P3、P4 和 P5 实例，然后再将您的 Amazon EKS 集群升级到版本 1.25。您还应主动升级您的应用程序以支持 NVIDIA 525 系列或更高版本。我们计划在 2024 年 1 月晚些时候将更新的 NVIDIA 525 系列或更新驱动程度反向移植到 Kubernetes 版本 1.23 和 1.24。
- 在版本 1.30 或更高版本中新创建的任何托管节点组都将自动默认使用 AL2023 作为节点操作系统。以前，新节点组将默认为 AL2。在创建新节点组时，您可以通过选择 AL2 作为 AMI 类型来继续使用 AL2。

- 对 AL2 的支持将于 2025 年 6 月 30 日结束。更多有关信息，请参阅 Amazon Linux 2 FAQs。

从 AL2 升级到 AL2023

Amazon EKS 优化的 AMI 有两个基于 AL2 和 AL2023 的系列可供选择。AL2023 是一款新的基于 Linux 的操作系统，旨在为您的云应用程序提供安全、稳定和高性能的环境。它是 Amazon Web Services 推出的下一代 Amazon Linux，适用于所有支持的 Amazon EKS 版本，包括扩展支持中的版本 `1.23` 和 `1.24`。基于 AL2023 的 Amazon EKS 加速 AMI 将在以后的某个日期推出。如果您有加速工作负载，则应继续使用 AL2 加速的 AMI 或 Bottlerocket。

AL2023 比 AL2 提供了多项改进。有关完整比较，请参阅《Amazon Linux 2023 用户指南》中的比较 AL2 和 Amazon Linux 2023。已在 AL2 中添加、升级和移除了多个程序包。强烈建议在升级之前使用 AL2023 测试您的应用程序。有关 AL2023 中所有程序包更改的列表，请参阅《Amazon Linux 2023 发行说明》中的 Amazon Linux 2023 的程序包更改。

除了这些更改之外，您还应了解以下事项：

- AL2023 引入了使用 YAML 配置架构的新节点初始化流程 `nodeadm`。如果您使用的是自行管理的节点组或带有启动模板的 AMI，则在创建新节点组时，现在需要明确提供其他集群元数据。以下是最低必需参数的示例，其中 `apiServerEndpoint`、`certificateAuthority` 和服务 `cidr` 是必需的：

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

  在 AL2 中，来自这些参数的元数据是从 Amazon EKS `DescribeCluster` API 调用中发现的。使用 AL2023，这种行为发生了变化，因为在大型节点纵向扩展期间，额外的 API 调用有节流风险。如果您使用的是没有启动模板的托管节点组或者正在使用 Karpenter，则此更改不会影响您。有关 `certificateAuthority` 和服务 `cidr` 的更多信息，请参阅《Amazon EKS API 参考》中的 `DescribeCluster`。

- 在 AL2023 中不支持所有受支持的 Amazon EKS 版本的 Docker。在 AL2 中，对 Docker 的支持已结束，已在 Amazon EKS 版本 1.24 或更高版本中移除。有关弃用的更多信息，请参阅 Amazon EKS 已结束对 Dockershim 的支持。
- AL2023 需要 Amazon VPC CNI 版本 1.16.2 或更高版本。
- 默认情况下，AL2023 需要 IMDSv2。IMDSv2 有多项益处，可助于改善安全状况。它使用面向会话的身份验证方法，需要在简单的 HTTP PUT 请求中创建密钥令牌才能启动会话。会话令牌的有效时间可以介于 1 秒到 6 小时之间。有关如何从 IMDSv1 转换到 IMDSv2 的更多信息，请参阅转换到使用实例元数据服务版本 2 和获取 IMDSv2 的全部优势并在 AWS 基础设施中禁用 IMDSv1。如果您想使用 IMDSv1，则您仍然可以通过使用实例元数据选项启动属性手动覆盖设置来做到这一点。

  > **Note**
  >
  > 对于 IMDSv2，托管节点组的默认跳数设置为 1。这意味着，容器无法使用 IMDS 访问节点的凭证。如果您需要容器访问节点的凭证，则您仍然可以通过手动覆盖自定义 Amazon EC2 启动模板中的 HttpPutResponseHopLimit，将其增加到 2 来做到这一点。或者，您可以使用 Amazon EKS Pod 身份来提供凭证，而不是 IMDSv2。

- AL2023 采用下一代统一控制组层次结构（cgroupv2）。cgroupv2 用于实施容器运行时，并由 systemd 实施。虽然 AL2023 仍然包含可以使用 cgroupv1 使系统运行的代码，但这不是建议或支持的配置。此配置将在未来的 Amazon Linux 主要版本中完全移除。
- eksctl 需要 eksctl 版本 0.176.0 或更高版本才能支持 AL2023。

对于以前存在的托管节点组，您可以执行就地升级或蓝/绿升级，具体取决于您所使用的启动模板的方式：

- 如果您在托管节点组中使用自定义 AMI，则可以通过交换启动模板中的 AMI ID 来执行就地升级。在执行此升级策略之前，您应确保您的应用程序和所有用户数据首先传输到 AL2023。
- 如果您将托管节点组与标准启动模板或未指定 AMI ID 的自定义启动模板一起使用，则需要使用蓝/绿策略升级。蓝/绿升级通常更复杂，需要创建一个全新的节点组，在其中指定 AL2023 作为 AMI 类型。然后，需要小心配置新的节点组，以确保来自 AL2 节点组的所有自定义数据都与新操作系统兼容。在您的应用程序中对新节点组进行测试和验证后，Pods 可以从旧节点组迁移到新的节点组。迁移完成之后，您就可以删除旧节点组。

如果您正在使用 Karpenter 并想使用 AL2023，则需要使用 AL2023 修改该 EC2NodeClass amiFamily 字段。默认情况下，偏差在 Karpenter 中启用。这意味着，amiFamily 字段更改后，Karpenter 将自动将您的 Worker 节点更新为最新的 AMI（如果有）。

Amazon EKS 优化版加速型 Amazon Linux AMI

> **Note**
>
> 基于 AL2023 的 Amazon EKS 加速 AMI 将在以后的某个日期推出。如果您有加速工作负载，则应继续使用 AL2 加速的 AMI 或 Bottlerocket。

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，以支持基于 GPU、Inferentia 和 Trainium 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`
- AWS Neuron 驱动程序

有关加速的 AMI 中包含的最新组件的列表，请参阅 GitHub 上的 `amazon-eks-ami` 版本。

> **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 AWS 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 AWS Neuron（ML 加速器）的工作负载

有关在 Amazon EKS 中使用 Neuron 的训练和推理工作负载的详细信息，请参阅以下参考资料：

- AWS Neuron 文档中的容器 - Kubernetes - 入门
- 在 GitHub 上的 AWS Neuron EKS 示例中进行训练
- 使用 AWS Inferentia 进行的 Machine Learning 推理

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。

1. 当 GPU 节点加入集群后，您必须在您的集群上应用 适用于 Kubernetes 的 NVIDIA 设备插件，以作为 DaemonSet 使用。将 *vX.X.X* 替换为您需要的 NVIDIA/k8s-device-plugin 版本，然后运行以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
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。将 *tag* 替换为您需要的 nvidia/cuda 标签。此清单会启动一个 NVIDIA CUDA 容器，该容器将在一个节点上运行 `nvidia-smi`。

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:tag
    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
```

示例输出如下。

```
Mon Aug  6 20:23:31 20XX
+-----------------------------------------------------------------------------+
| NVIDIA-SMI XXX.XX                 Driver Version: XXX.XX                    |
|-------------------------------+----------------------+----------------------+
| 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。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。
- 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。

测试从 Docker 到 **containerd** 的迁移

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 结束了对 Dockershim 的支持。

对于 Kubernetes 版本 1.23，您可以使用可选的引导标志，为 Amazon EKS 优化版 AL2 AMI 启用 containerd 运行时系统。在更新到版本 1.24 或更高版本时，该功能提供迁移到 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。

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 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，其中包含以下内容。请将每个 *example value* 替换为您自己的值。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。要检索 ami-*1234567890abcdef0* 的优化版 AMI ID，请参阅 检索 Amazon EKS 优化版 Amazon Linux AMI ID。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
  version: 1.23
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。

运行以下命令以创建节点组。

```
eksctl create nodegroup -f my-nodegroup.yaml
```

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，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
```

更多信息

有关使用 Amazon EKS 优化版 Amazon Linux AMI 的更多信息，请参阅以下部分：

- 要将 Amazon Linux 与托管节点组一起使用，请参阅 托管节点组。
- 要启动自行管理的 Amazon Linux 节点，请参阅 检索 Amazon EKS 优化版 Amazon Linux AMI ID。
- 有关版本信息，请参阅 Amazon EKS 优化版 Amazon Linux AMI 版本。
- 要检索 Amazon EKS 优化版 Amazon Linux AMI 的最新 ID，请参阅 检索 Amazon EKS 优化版 Amazon Linux AMI ID。
- 要获取用于构建 Amazon EKS 优化版 AMI 的开源脚本，请参阅 Amazon EKS 优化版 Amazon Linux AMI 生成脚本。

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
```

每个 AMI 版本都包含 kubelet、Docker、Linux 内核及 containerd 的各种版本。加速 AMI 还包括 NVIDIA 驱动程序的各种版本。您可以在 GitHub 上的更改日志中找到该版本。

检索 Amazon EKS 优化版 Amazon Linux AMI ID

您可以使用编程方式，通过查询 AWS 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。

使用 AWS CLI 检索适用于 Amazon EKS 优化 AMI 的 AMI ID

1. 确定您的节点实例将部署到哪个区域，例如 us-west-2。
2. 确定您需要的 AMI 类型。有关 Amazon EC2 实例类型的更多信息，请参阅实例类型。
   - amazon-linux-2 适用于基于 Amazon Linux 2（AL2）x86 的实例。
   - amazon-linux-2-arm64 适用于 AL2 ARM 实例，例如基于 AWS Graviton 实例。
   - amazon-linux-2-gpu 适用于 AL2 GPU 加速实例。
   - amazon-linux-2023/x86_64/standard 适用于基于 Amazon Linux 2023（AL2023）x86 的实例。
   - amazon-linux-2023/arm64/standard 适用于 AL2023 ARM 实例。
3. 确定您的节点将会附加到集群的 Kubernetes 版本，例如 1.30。
4. 运行以下 AWS CLI 命令以检索合适的 AMI ID。根据需要替换 AWS 区域、Kubernetes 版本和平台。您必须使用具有 ssm:GetParameter IAM 权限的 IAM 主体登录，才能检索 Amazon EKS 优化 AIM 元数据。

   ```
   aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.30/amazon-linux-2/
   recommended/image_id \
                   --region region-code --query "Parameter.Value" --output text
   ```

   示例输出如下。

   ```
   ami-1234567890abcdef0
   ```

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（AL2）和 Amazon Linux 2023（AL2023）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 AWS IAM 身份验证器以及从头开始构建您自己的基于 Amazon Linux 的 AMI 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和 nodeadm 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 AWS 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 优化版 AL2 包含可选的引导标记，用于启用 `containerd` 运行时。

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（AL2）、Ubuntu 18 和 Ubuntu 20 的 VT1 视频转码实例系列。VT1 支持带有加速 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 AL2、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 Amazon EKS 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（AL2）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 中。
- 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。
- Helm 3.x
- 用于安装 MPI Operator 的 Helm Chart。
- MPI Operator

1. 从 AL2、Ubuntu 18 或 Ubuntu 20 创建并启动基本 AMI。
2. 按照这些说明以为 DL1 设置环境。

Amazon EKS 优化版 Bottlerocket AMI

Bottlerocket 是由 AWS 赞助和支持的开源 Linux 发行版。Bottlerocket 专为托管容器工作负载而构建。借助 Bottlerocket，您可以通过实现容器基础设施自动更新来提高容器化部署的可用性并降低运营成本。Bottlerocket 仅包含运行容器所需的必备软件，可提高资源利用率、减少安全威胁并降低管理开销。Bottlerocket AMI 包括 `containerd`、`kubelet` 和 AWS IAM 身份验证器。除了托管节点组和自行管理的节点外，Bottlerocket 还受 Karpenter 支持。

优势

将 Bottlerocket 与 Amazon EKS 集群结合使用有以下优势：

- 正常运行时间更长，运营成本更低，管理复杂性 – 与其他 Linux 发行版相比，Bottlerocket 资源占用量更小，启动时间更短，更不容易受到安全威胁。Bottlerocket's 占用量更小，因此可通过使用更少的存储、计算和网络资源来帮助用户降低成本。
- 自动操作系统更新可提高安全性 – Bottlerocket 的更新作为单个单元应用，必要时可以回滚。这消除了更新损坏或失败的风险，这些情况可能会使系统处于不可用状态。借助 Bottlerocket，安全更新能在可用时以中断最少的方式自动应用，并在出现故障时回滚。
- Premium Support：AWS 在 Amazon EC2 上提供的 Bottlerocket 发行版包含在相同的 AWS Support 计划中，这些计划还涵盖 Amazon EC2、Amazon EKS 和 Amazon ECR 等 AWS 服务。

注意事项

将 Bottlerocket 用于 AMI 类型时，请考虑以下事项：

- Bottlerocket 支持采用 x86_64 和 arm64 处理器的 Amazon EC2 实例。Bottlerocket 不建议将 Amazon EC2 实例与 Inferentia 芯片一起使用。
- 目前还没有可用于部署 Bottlerocket 节点的 AWS CloudFormation 模板。
- Bottlerocket 映像不附带 SSH 服务器或 Shell。您可以使用带外访问方法来允许 SSH。这些方法会启用管理员容器，并利用用户数据来执行一些引导配置步骤。有关更多信息，请参阅 GitHub 上 Bottlerocket 操作系统 一文中的以下章节：
  - Exploration (探索)
  - 管理员容器
  - Kubernetes 设置
- Bottlerocket 使用不同的容器类型：
  - 默认情况下，将启用控制容器。该容器运行 AWS 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 AWS 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，则会启用管理员容器。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。有关更多信息，请参阅 GitHub 上的 Admin 容器。

更多信息

有关使用 Amazon EKS 优化版 Bottlerocket AMI 的更多信息，请参阅以下部分：

- 有关 Bottlerocket 的详细信息，请参阅 GitHub 上的文档和发布。
- 要将 Bottlerocket 与托管节点组一起使用，请参阅 托管节点组。
- 要启动自行管理的 Bottlerocket 节点，请参阅 启动自行管理的 Bottlerocket 节点。
- 要检索 Amazon EKS 优化版 Bottlerocket AMI 的最新 ID，请参阅 检索 Amazon EKS 优化版 Bottlerocket AMI ID。
- 有关合规支持的详细信息，请参阅 Bottlerocket 合规支持。

检索 Amazon EKS 优化版 Bottlerocket AMI ID

您可以通过查询 AWS 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。您使用的 IAM 主体必须具有 `ssm:GetParameter` IAM 权限才能检索 Amazon EKS 优化版 AMI 元数据。

您可以使用子参数 `image_id`，通过以下 AWS CLI 命令检索推荐的最新 Amazon EKS 优化版 Bottlerocket AMI 的镜像 ID。请将 *1.30* 替换为支持的版本，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-1.30/x86_64/latest/
image_id --region region-code --query "Parameter.Value" --output text
```

示例输出如下。

```
ami-1234567890abcdef0
```

Bottlerocket 合规支持

Bottlerocket 符合多个组织制定的建议：

- 相关人士定义了适用于 Bottlerocket 的 CIS Benchmark。在默认配置中，Bottlerocket 映像具有 CIS 1 级配置配置文件所要求的大部分控件。您可以实现 CIS 2 级配置配置文件所要求的控制。有关更多信息，请参阅 AWS 博客上的根据 CIS Benchmark 验证 Amazon EKS 优化版 Bottlerocket AMI。
- 经优化的功能集和更小的攻击面意味着 Bottlerocket 实例需要更少的配置便可满足 PCI DSS 要求。适用于 Bottlerocket 的 CIS Benchmark 是巩固指南的理想资源，可支持您达到要求，实现符合 PCI DSS 要求 2.2 规定的安全配置标准。您还可以利用 Fluent Bit 来支持您达到要求，实现符合 PCI DSS 要求 10.2 规定的操作系统级别审核日志。AWS 定期发布新的（经修补的）Bottlerocket 实例，以帮助您满足 PCI DSS 要求 6.2（适用于 v3.2.1）和要求 6.3.3（适用于 v4.0）。
- Bottlerocket 是一项符合 HIPAA 要求的功能，已获授权用于Amazon EC2 和 Amazon EKS 的受监管工作负载。有关更多信息，请参阅 Amazon EKS 上的 HIPAA 安全性和合规性架构设计白皮书。

Amazon EKS 优化版 Ubuntu Linux AMI

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 AWS 联合开发的自定义 AWS 内核。有关更多信息，请参阅Amazon Elastic Kubernetes Service（EKS）上的 Ubuntu 和 启动自行管理的 Ubuntu 节点。有关支持的更多信息，请参阅 AWS Premium Support 常见问题解答中的第三方软件部分。

Amazon EKS 优化版 Windows AMI

Windows Amazon EKS 优化版 AMI 是基于 Windows Server 2019 和 Windows Server 2022 构建的。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`
- 适用于 Kubernetes 的 AWS IAM 身份验证器
- `csi-proxy`
- `containerd`

> **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
- Amazon EKS 优化版 Windows Server 2022 Core AMI
- Amazon EKS 优化版 Windows Server 2022 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。
> - 为了确保您在默认情况下安装了最新的安全更新，Amazon EKS 将维持最近 4 个月的经优化的 Windows AMI。自首次发布之日起，每个新 AMI 的可用期为 4 个月。在此期限之后，较旧的 AMI 将变为私有且不能再访问。我们鼓励使用最新的 AMI，以避免出现安全漏洞，避免

无法访问已达到其支持生命周期尽头的旧 AMI。虽然我们不能保证可以提供对已设为私有的 AMI 的访问权限，但您可以通过向 AWS Support 提交服务单来请求访问权限。

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 2022 Core	10/17/2022	
Windows Server 2022 Full	10/17/2022	
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。您可以通过将自定义参数值指定为引导脚本的参数。例如，您可以更新启动模板中的用户数据。有关更多信息，请参阅 Amazon EC2 用户数据。

此脚本包含以下命令行参数：

- `-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`。
- `-ServiceCIDR` – 覆盖从中寻址集群服务的 Kubernetes 服务 IP 地址范围。基于主接口的 IP 地址，默认值为 `172.20.0.0/16` 或 `10.100.0.0/16`。
- `-ExcludedSnatCIDRs` – 要从源网络地址转换（SNAT）中排除的 `IPv4` CIDR 列表。这意味着，VPC 可寻址的容器组（pod）私有 IP 不会转换为用于出站流量的实例 ENI 主 `IPv4` 地址的 IP 地址。默认情况下，系统会添加 Amazon EKS Windows 节点的 VPC 的 `IPv4` CIDR。为该参数指定 CIDR 还会另外排除所指定的 CIDR。有关更多信息，请参阅 适用于 Pods 的 SNAT。

除了命令行参数之外，您还可以指定一些环境变量参数。指定命令行参数时，它优先于相应的环境变量。环境变量应定义为机器（或系统）作用范围，因为引导脚本只会读取机器范围的变量。

该脚本考虑以下环境变量：

- `SERVICE_IPV4_CIDR` — 有关定义，请参阅 `ServiceCIDR` 命令行参数。
- `EXCLUDED_SNAT_CIDRS` — 应为以逗号分隔的字符串。有关定义，请参阅 `ExcludedSnatCIDRs` 命令行参数。

通过 **`eksctl`** 启动自行管理的 Windows Server 2022 节点

您可以使用以下 **`test-windows-2022`**`.yaml` 作为参考，将 Windows Server 2022 作为自行管理的节点运行。请将每个 *`example value`* 替换为您自己的值。

> **Note**
>
> 您必须使用 `eksctl` 版本 `0.116.0` 或更高版本来运行自行管理的 Windows Server 2022 节点。

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
```

```
  region: region-code
  version: '1.30'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - 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-2022.yaml
```

gMSA 身份验证支持

Amazon EKS Windows Pods 允许不同类型的群组托管服务账户（gMSA）身份验证。

- Amazon EKS 支持使用 Active Directory 域身份进行身份验证。有关加入域的 gMSA 的更多信息，请参阅 AWS 博客上的 Amazon EKS Windowspods 上的 Windows 身份验证。
- Amazon EKS 提供一个插件，使未加入域的 Windows 节点能够使用可移植用户身份检索 gMSA 凭证。有关无域 gMSA 的更多信息，请参阅 AWS 博客上的 Amazon EKS Windowspods 上的无域 Windows 身份验证。

缓存的容器映像

Amazon EKS Windows 优化版 AMI 缓存了某些容器映像，用于 `containerd` 运行时。使用 Amazon 托管的构建组件构建自定义 AMI 时，系统会缓存容器映像。有关更多信息，请参阅 使用 Amazon 托管的构建组件。

以下缓存的容器映像适用于 `containerd` 运行时系统：

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`

- `mcr.microsoft.com/windows/servercore`

更多信息

有关使用 Amazon EKS 优化版 Windows AMI 的更多信息，请参阅以下部分：

- 要将 Windows 与托管节点组一起使用，请参阅 托管节点组。
- 要启动自行管理的 Windows 节点，请参阅 启动自行管理的 Windows 节点。
- 有关版本信息，请参阅 Amazon ECS 优化版 Windows AMI 版本。
- 要检索 Amazon EKS 优化版 Windows AMI 的最新 ID，请参阅 检索 Amazon EKS 优化版 Windows AMI ID。
- 要使用 Amazon EC2 Image Builder 创建自定义 Amazon EKS 优化版 Windows AMI，请参阅 创建自定义 Amazon EKS 优化版 Windows AMI。
- 有关最佳实践，请参阅《EKS 最佳实践指南》中的经 Amazon EKS 优化的 Windows AMI 管理。

Amazon ECS 优化版 Windows AMI 版本

> **Important**
>
> AWS 发布的 Amazon EKS 优化版 Windows AMI 的延期支持不适用于 Kubernetes 版本 `1.23`，但适用于 Kubernetes 版本 `1.24` 及更高版本。

本主题列出了 Amazon EKS 优化版 Windows AMI 的版本及其 `kubelet`、`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。

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
```

> **Note**
>
> Amazon EKS 托管节点组支持 2022 年 11 月及更高版本的 Windows AMI。

Amazon EKS 优化版 Windows Server 2022 Core AMI

以下各表列出了 Amazon EKS 优化版 Windows Server 2022 Core AMI 的当前版本和以前版本。

Kubernetes version 1.30

Kubernetes 版本 **1.30**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes 版本 **1.29**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已将 containerd 升级到 1.7.11。已将 kubelet 升级到 1.29.3。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	修复了以下错误：kubelet 垃圾收集过程错误删除了暂停图像。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	不包括 Windows Server 2022 Core AMI 上的独立 Windows 更新 KB5034439。知识库仅适用于具有单独 WinRE 分区的 Windows 安装，不包含在我们的任何 Amazon EKS 优化版 Windows AMI 中。

Kubernetes version 1.28

Kubernetes 版本 **1.28**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	不包括 Windows Server 2022 Core AMI 上的独立 Windows 更新 KB5034439。知识库仅

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
				适用于具有单独 WinRE 分区的 Windows 安装，不包含在我们的任何 Amazon EKS 优化版 Windows AMI 中。
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	修复了 kubelet 中的安全通告。
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 版本 **1.27**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.27.12。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.11	1.27.9	1.6.18	1.1.2	不包括 Windows Server 2022 Core AMI 上的独立 Windows 更新 KB5034439。知识库仅适用于具有单独 WinRE 分区的 Windows 安装，不包含在我们的任何 Amazon EKS 优化版 Windows AMI 中。
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	修复了 kubelet 中的安全通告。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	已修复 containers-roadmap 问题 #2042，该问题导致节点无法提取私有 Amazon ECR 镜像。
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 版本 **1.26**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.26.15。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.11	1.26.12	1.6.18	1.1.2	不包括 Windows Server 2022 Core AMI 上的独立 Windows 更新 KB5034439。知识库仅适用于具有单独 WinRE 分区的 Windows 安装，不包含在我们的任何 Amazon EKS 优化版 Windows AMI 中。
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.26.9。添加了新的引导脚本环

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
				境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	已将 Kubernetes 升级到 1.26.4。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 版本 **1.25**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.11	1.25.16	1.6.18	1.1.2	不包括 Windows Server 2022 Core AMI 上的独立 Windows 更新 KB5034439。知识库仅适用于具有单独 WinRE 分区

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
				的 Windows 安装，不包含在我们的任何 Amazon EKS 优化版 Windows AMI 中。
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.25.14。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	已将 Kubernetes 升级到 1.25.9。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 版本 **1.24**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.11	1.24.17	1.6.18	1.1.2	不包括 Windows Server 2022 Core AMI 上的独立 Windows 更新 KB5034439。知识库仅适用于具有单独 WinRE 分区的 Windows 安装，不包含在我们的任何 Amazon EKS 优化版 Windows AMI 中。
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.24.17。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	已将 Kubernetes 升级到 1.24.13。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes 版本降级为 1.24.7，因为 1.24.10 具有 kube-proxy 中报告的问题。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

Amazon EKS 优化版 Windows Server 2022 Full AMI

以下各表列出了 Amazon EKS 优化版 Windows Server 2022 Full AMI 的当前版本和以前版本。

Kubernetes version 1.30

Kubernetes 版本 **1.30**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes 版本 **1.29**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已将 containerd 升级到 1.7.11。已将 kubelet 升级到 1.29.3。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	修复了以下错误：kubelet 垃圾收集过程错误删除了暂停图像。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes 版本 **1.28**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	修复了 kubelet 中的安全通告。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 版本 **1.27**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.27.12。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	修复了 kubelet 中的安全通告。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	已修复 containers-roadmap 问题 #2042，该问题导致节点无法提取私有 Amazon ECR 镜像。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 版本 **1.26**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.26.15。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.26.9。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	已将 Kubernetes 升级到 1.26.4。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 版本 **1.25**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.25.14。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	

AMI 版本	**kubelet** 版本	**containe d** 版本	**csi-proxy** 版本	发布说明
1.25-2023 .06.20	1.25.9	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.25-2023 .06.14	1.25.9	1.6.6	1.1.1	已将 Kubernetes 升级到 1.25.9。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.25-2023 .05.09	1.25.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedS natCIDRs ）。
1.25-2023 .04.11	1.25.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.25-2023 .03.27	1.25.6	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。
1.25-2023 .03.20	1.25.6	1.6.6	1.1.1	
1.25-2023 .02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 版本 **1.24**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.24.17。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
				文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	已将 Kubernetes 升级到 1.24.13。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes 版本降级为 1.24.7，因为 1.24.10 具有 kube-proxy 中报告的问题。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

Amazon EKS 优化版 Windows Server 2019 Core AMI

以下各表列出了 Amazon EKS 优化版 Windows Server 2019 Core AMI 的当前版本和以前版本。

Kubernetes version 1.30

Kubernetes 版本 **1.30**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes 版本 **1.29**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已将 containerd 升级到 1.7.11。已将 kubelet 升级到 1.29.3。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	修复了以下错误：kubelet 垃圾收集过程错误删除了暂停图像。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes 版本 **1.28**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	修复了 kubelet 中的安全通告。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 版本 **1.27**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.27.12。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	修复了 kubelet 中的安全通告。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	已修复 containers-roadmap 问题 #2042，该问题导致节点无法提取私有 Amazon ECR 镜像。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
11.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 版本 **1.26**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.26.15。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.26.9。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	已将 Kubernetes 升级到 1.26.4。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 版本 **1.25**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.25.14。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	已将 Kubernetes 升级到 1.25.9。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 版本 **1.24**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.24.17。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
				文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	已将 Kubernetes 升级到 1.24.13。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes 版本降级为 1.24.7，因为 1.24.10 具有 kube-proxy 中报告的问题。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.11.08	1.24.7	1.6.6	1.1.1	

Amazon EKS 优化版 Windows Server 2019 Full AMI

以下各表列出了 Amazon EKS 优化版 Windows Server 2019 Full AMI 的当前版本和以前版本。

Kubernetes version 1.30

Kubernetes 版本 **1.30**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes version 1.29

Kubernetes 版本 **1.29**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	已将 containerd 升级到 1.7.11。已将 kubelet 升级到 1.29.3。
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	修复了以下错误：kubelet 垃圾收集过程错误删除了暂停图像。
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes 版本 **1.28**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.28.8。
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	修复了 kubelet 中的安全通告。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 版本 **1.27**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2024.05.14	1.27.12	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.27.12。
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	修复了 kubelet 中的安全通告。
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	已修复 containers-roadmap 问题 #2042，该问题导致节点无法提取私有 Amazon ECR 镜像。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 版本 **1.26**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2024.05.14	1.26.15	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。已将 kubelet 升级到 1.26.15。
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.26.9。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	已将 Kubernetes 升级到 1.26.4。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 版本 **1.25**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.05.14	1.25.16	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy 。
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.25.14。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS ）。
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	已将 Kubernetes 升级到 1.25.9。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 版本 **1.24**

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2024.05.14	1.24.17	1.6.28	1.1.2	已将 containerd 升级到 1.6.28。
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	已将 containerd 升级到 1.6.25。已使用 golang 1.22.1 重建 CNI 和 csi-proxy。
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	包括 CVE-2023-5528 的补丁。
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	已将 containerd 升级到 1.6.18。已将 kubelet 升级到 1.24.17。添加了新的引导脚本环境变量（SERVICE_IPV4_CIDR 和 EXCLUDED_SNAT_CIDRS）。
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	已将 Amazon VPC CNI 插件升级为使用 Kubernetes 连接器二进制

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
				文件，后者从 Kubernetes API 服务器获取 Pod IP 地址。合并的拉取请求 #100。
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	包括 CVE-2023-3676 、CVE-2023-3893 和 CVE-2023-3955 的补丁。
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.21	1.24.13	1.6.6	1.1.1	解决了导致 DNS 后缀搜索列表填充不正确的问题。
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	已将 Kubernetes 升级到 1.24.13。在 CNI 中增加了对主机端口映射的支持。合并的拉取请求 #93。
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	修复了节点重启后导致容器组（pod）上出现网络连接问题 #1126 的错误。引入了新的引导脚本配置参数（ExcludedSnatCIDRs ）。
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	针对服务崩溃为 kubelet 和 kube-proxy 添加了恢复机制。
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	安装了无域 gMSA 插件以促进 Amazon EKS 上 Windows 容器的 gMSA 身份验证。

AMI 版本	**kubelet** 版本	**containerd** 版本	**csi-proxy** 版本	发布说明
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes 版本降级为 1.24.7，因为 1.24.10 具有 kube-proxy 中报告的问题。
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.12	1.24.7	1.6.6	1.1.1	

检索 Amazon EKS 优化版 Windows AMI ID

您可以使用编程方式，通过查询 AWS 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。您使用的 IAM 主体必须具有 `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 *1.30* 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。对于 Kubernetes `1.24` 版本或更高版本，您可以将 *2019* 替换为 `2022` 以查看 Windows Server 2022 AMI ID。

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2019-
English-Core-EKS_Optimized-1.30/image_id --region region-code --query "Parameter.Value"
 --output text
```

示例输出如下。

```
ami-1234567890abcdef0
```

创建自定义 Amazon EKS 优化版 Windows AMI

您可以通过以下任一方式，使用 EC2 Image Builder 创建自定义 Amazon EKS 优化版 Windows AMI：

- 使用 Amazon EKS 优化版 Windows AMI 作为基础
- 使用 Amazon 托管的构建组件

无论采用哪种方式，您必须创建自己的 Image Builder 配方。有关更多信息，请参阅《Image Builder 用户指南》中的创建映像配方的新版本。

> **Important**
>
> 以下 Amazon 托管的 eks 组件包括 CVE-2023-5528 的补丁。
>
> - 1.24.3 和更高版本
> - 1.25.2 和更高版本
> - 1.26.2 和更高版本
> - 1.27.0 和更高版本
> - 1.28.0 和更高版本

使用 Amazon EKS 优化版 Windows AMI 作为基础

推荐使用该选项构建自定义 Windows AMI。我们提供的 Amazon EKS 优化版 Windows AMI 比 Amazon 托管的构建组件更新得更频繁。

1. 开启新的 Image Builder 配方。
   a. 打开位于 https://console.aws.amazon.com/imagebuilder的 EC2 Image Builder 控制台

b. 在左侧导航窗格中，选择映像配方。

c. 选择创建映像配方。

2. 在配方详细信息部分中，输入名称和版本。

3. 在基础映像部分指定 Amazon EKS 优化版 Windows AMI 的 ID。

a. 选择输入自定义 AMI ID。

b. 检索所需 Windows 操作系统版本的 AMI ID。有关更多信息，请参阅检索 Amazon EKS 优化版 Windows AMI ID。

c. 输入自定义 AMI ID。如果找不到 AMI ID，请确保 AMI ID 的 AWS 区域 与控制台右上角显示的 AWS 区域 相匹配。

4. （可选）要获取最新的安全更新，请在构建组件 - 部分中添加 `update-windows` 组件。

a. 在按名称查找组件搜索框右侧的下拉列表中，选择 Amazon 托管。

b. 在按名称查找组件搜索框中，输入 **`update-windows`**。

c. 选中 **`update-windows`** 搜索结果的复选框。此组件包括操作系统的最新 Windows 补丁。

5. 使用所需配置完成剩余的映像配方输入。有关更多信息，请参阅《Image Builder 用户指南》中的创建新的映像配方版本（控制台）。

6. 选择创建配方。

7. 在新的或现有的映像管道中使用新的映像配方。映像管道成功运行后，您的自定义 AMI 将作为输出映像列出并随时可用。有关更多信息，请参阅使用 EC2 Image Builder 控制台向导创建映像管道。

使用 Amazon 托管的构建组件

当使用 Amazon EKS 优化版 Windows AMI 作为基本操作系统不可行时，您可以改用 Amazon 托管的构建组件。此选项可能落后于支持的最新 Kubernetes 版本。

1. 开启新的 Image Builder 配方。

a. 打开位于 https://console.aws.amazon.com/imagebuilder的 EC2 Image Builder 控制台

b. 在左侧导航窗格中，选择映像配方。

c. 选择创建映像配方。

2. 在配方详细信息部分中，输入名称和版本。

3. 在基础映像部分中确定您将使用哪个选项来创建自定义 AMI：

- 选择托管映像 – 为您的映像操作系统（OS）选择 Windows。然后，为映像来源选择以下选项之一。
  - 快速入门（Amazon 托管）– 在映像名称下拉列表中，选择一个 Amazon EKS 支持的 Windows Server 版本。有关更多信息，请参阅Amazon EKS 优化版 Windows AMI。
  - 我拥有的映像 – 对于映像名称，请选择具有自带许可证的自有映像的 ARN。您提供的镜像尚未安装 Amazon EKS 组件。
- 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。

4. 在构建组件 – Windows 部分中，执行以下操作：
   a. 在按名称查找组件搜索框右侧的下拉列表中，选择 Amazon 托管。
   b. 在按名称查找组件搜索框中，输入 **eks**。
   c. 选中 **eks-optimized-ami-windows** 搜索结果的复选框，即使返回的结果可能不是您想要的版本。
   d. 在按名称查找组件搜索框中，输入 **update-windows**。
   e. 选中 update-windows 搜索结果的复选框。此组件包括操作系统的最新 Windows 补丁。
5. 在选定的组件部分中，执行以下操作：
   a. 选择 **eks-optimized-ami-windows** 的版本控制选项。
   b. 选择指定组件版本。
   c. 在组件版本字段中，输入 ***version.x*** ，将 *version* 替换为支持的 Kubernetes 版本。在版本号部分输入 *x* 表示使用与您明确定义的版本相应部分也相符的最新组件版本。注意控制台输出，因为它会就您所需版本是否作为托管组件提供给出建议。请记住，最新 Kubernetes 版本可能不适用于构建组件。有关可用版本的更多信息，请参阅 检索有关 eks-optimized-ami-windows 组件版本的信息。

   > **Note**
   >
   > 以下 `eks-optimized-ami-windows` 构建组件版本需要 `eksctl` 版本 `0.129` 或更低版本：
   >
   > - `1.24.0`

6. 使用所需配置完成剩余的映像配方输入。有关更多信息，请参阅《Image Builder 用户指南》中的创建新的映像配方版本（控制台）。

7. 选择创建配方。
8. 在新的或现有的映像管道中使用新的映像配方。映像管道成功运行后，您的自定义 AMI 将作为输出映像列出并随时可用。有关更多信息，请参阅使用 EC2 Image Builder 控制台向导创建映像管道。

检索有关 **eks-optimized-ami-windows** 组件版本的信息

您可以检索有关每个组件所安装内容的特定信息。例如，您可以验证安装了哪个 `kubelet` 版本。这些组件在 Amazon EKS 支持的 Windows 操作系统版本上进行功能测试。有关更多信息，请参阅发布日历。未列为受支持或已进入终止支持状态的任何其他 Windows OS 版本可能与组件不兼容。

1. 打开位于 https://console.aws.amazon.com/imagebuilder的 EC2 Image Builder 控制台
2. 在左侧导航窗格中选择组件。
3. 在按名称查找组件搜索框右侧的下拉列表中，将我拥有的更改为快速入门（Amazon 托管）。
4. 在 Find components by name（按名称查找组件）框中，输入 **eks**。
5. （可选）如果您使用的是最新版本，请选择两次，按降序对版本列进行排序。
6. 选择具有所需版本的 **eks-optimized-ami-windows** 链接。

结果页面中的描述显示了具体信息。

存储

本章介绍 Amazon EKS 集群的存储选项。

主题

Amazon EBS CSI 驱动程序

Amazon Elastic Block Store（Amazon EBS）Container Storage Interface（CSI）驱动程序管理作为您所创建 Kubernetes 卷存储的 Amazon EBS 卷的生命周期。Amazon EBS CSI 驱动程序为以下类型的 Kubernetes 卷制作 Amazon EBS 卷：通用临时卷和持久性卷。

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 AWS API。有关更多信息，请参阅 创建 Amazon EBS CSI 驱动程序 IAM 角色。
- 您无法将 Amazon EBS 卷挂载到 Fargate Pods。
- 您可以在 Fargate 节点上运行 Amazon EBS CSI 控制器，但要在 Amazon EBS CSI 节点 DaemonSet 只能在 Amazon EC2 实例上运行。

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。

- 有关如何将其添加为自行管理安装的说明，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。

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 AWS API。有关更多信息，请参阅 GitHub 上的设置驱动程序权限。

> **Note**
>
> Pods 将有权访问分配给 IAM 角色的权限，除非您阻止对 IMDS 的访问。有关更多信息，请参阅 Amazon EKS 的安全最佳实践。

先决条件

- 现有集群。
- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。

以下过程为您演示了如何创建 IAM 角色并向其附加 AWS 托管策略。您可以使用 `eksctl`、AWS Management Console或 AWS CLI。

> **Note**
>
> 此过程中的具体步骤是为将驱动程序用作 Amazon EKS 附加组件而编写的。要将驱动程序用作自行管理的附加组件，需要遵循不同的步骤。有关更多信息，请参阅 GitHub 上的设置驱动程序权限。

eksctl

使用 **`eksctl`** 创建 Amazon EBS CSI 插件 IAM 角色

1. 创建 IAM 角色并附加到策略。AWS 维护 AWS 托管策略，或者您可创建自己的自定义策略。您可以使用以下命令创建 IAM 角色并将 AWS 托管策略附加到其上。将 *`my-cluster`* 替换为您的集群名称。此命令将部署 AWS CloudFormation 堆栈，该堆栈将创建 IAM 角色，并会将 IAM policy 附加到该堆栈。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
    --approve
```

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-arn`* 替换为自定义 KMS 密钥 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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
```

AWS Management Console

使用 AWS Management Console 创建 Amazon EBS CSI 插件 IAM 角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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`。
d. 选择下一步。

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. 在添加标签（可选）下，将标签作为键值对附加，以将元数据添加到角色。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 资源。
   c. 选择创建角色。
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/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

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 AWS 区域。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为集群的 OIDC 提供商 ID。

```
"oidc.eks.region-code.amazonaws.com/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，并将 *custom-key-arn* 替换为自定义 KMS 密钥 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

e. 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. 从添加权限下拉列表中，选择附加策略。

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. 选择附加策略。

AWS CLI

使用 AWS 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
```

示例输出如下。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. 创建 IAM 角色并向其授予 AssumeRoleWithWebIdentity 操作。

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 *EXAMPLED539D4633E53DE1B71EXAMPLE* 和 *region-code* 替换为在上一步中返回的值。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 arn:aws: 替换为 arn:aws-us-gov:。

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

```
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/
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. 附加策略。AWS 维护 AWS 托管策略，或者您可创建自己的自定义策略。使用以下命令以将 AWS 托管策略附加到角色。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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-arn* 替换为自定义 KMS 密钥 ARN。

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

```
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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
```

现在您已经创建了 Amazon EBS CSI 驱动程序 IAM 角色，您可以继续前往 添加 Amazon EBS CSI 驱动程序附加组件。在该过程中部署插件后，系统会创建并配置为使用名为 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。

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。您可以按照 添加 Amazon EBS CSI 驱动程序附加组件 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 和 删除 Amazon EBS CSI 附加组件 部分的步骤进行管理。

先决条件

- 现有集群。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 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- Amazon EBS CSI 驱动程序 IAM 角色。如果您未满足此先决条件，则尝试安装附加组件并运行 kubectl describe pvc 时，将显示 failed to provision volume with StorageClass 和 could not create volume in EC2: UnauthorizedOperation 错误。有关更多信息，请参阅创建 Amazon EBS CSI 驱动程序 IAM 角色。
- 如果您使用的是集群范围内受限的 PodSecurityPolicy，请确保授予该附加组件足够的权限，以进行部署。有关每个附加组件 Pod 所需的权限，请参阅 GitHub 上的相关附加组件清单定义。

> **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 驱动程序附加组件

> **Important**
>
> 在将 Amazon EBS 驱动程序添加为 Amazon EKS 附加组件之前，请确认您的集群上没有安装该驱动程序的自行管理版本。如果安装了自行管理版本，请参阅 GitHub 上的 卸载自行管理的 Amazon EBS CSI 驱动程序。

您可以使用 `eksctl`、AWS Management Console 或 AWS 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。将 *`my-cluster`* 替换为您的集群的名称，将 *`111122223333`* 替换为您的账户 ID，并将 *`AmazonEKS_EBS_CSI_DriverRole`* 替换为之前创建的 IAM 角色。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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。有关此设置其他选项的更多信息，请参阅 `eksctl` 文档中的 Addons（附加组件）。有关 Amazon EKS Kubernetes 字段管理的更多信息，请参阅 Kubernetes 字段管理。

AWS Management Console

要使用 AWS 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. 选择附加组件选项卡。
5. 选择获取更多附加组件。
6. 在选择附加组件页面上，执行以下操作：
   a. 在Amazon EKS 附加组件部分，选择 Amazon EBS CSI Driver 复选框。
   b. 选择下一步。
7. 在配置选定的附加组件设置页面上，执行以下操作：
   a. 选择您想使用的 Version（版本）。
   b. 对于选择 IAM 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. （可选）展开可选配置设置。如果对冲突解决方法选择覆盖，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。
   d. 选择下一步。
8. 在查看和添加页面上，选择创建。附加组件安装完成后，您将看到已安装的附加组件。

AWS CLI

要使用 AWS 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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
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
```

现在您已经将 Amazon EBS CSI 驱动程序添加为 Amazon EKS 附加组件，可以继续 部署示例应用程序并验证 CSI 驱动程序是否正常运行。此过程包括设置存储类。

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。

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
```

示例输出如下。

```
NAME                    VERSION                 STATUS  ISSUES  IAMROLE
 UPDATE AVAILABLE
aws-ebs-csi-driver      v1.11.2-eksbuild.1      ACTIVE  0
  v1.11.4-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 v1.11.4-eksbuild.1 --
cluster my-cluster \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --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。有关此设置其他选项的更多信息，请参阅 eksctl 文档中的 Addons（附加组件）。有关 Amazon EKS Kubernetes 字段管理的更多信息，请参阅 Kubernetes 字段管理。

AWS Management Console

使用 AWS 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. 选择附加组件选项卡。
5. 选择 Amazon EBS CSI 驱动程序。
6. 选择编辑。
7. 在配置 Amazon EBS CSI 驱动程序页面上，执行以下操作：
   a. 选择您想使用的 Version（版本）。
   b. 对于选择 IAM 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. （可选）展开可选配置设置并根据需要进行修改。
   d. 选择保存更改。

AWS CLI

使用 AWS 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
```

示例输出如下。

```
v1.11.2-eksbuild.1
```

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 1.23 \
  --query "addons[].addonVersions[].[addonVersion,
 compatibilities[].defaultVersion]" --output text
```

示例输出如下。

```
v1.11.4-eksbuild.1
True
```

```
v1.11.2-eksbuild.1
False
```

下方使用 True 的版本是创建附加组件时部署的默认版本。创建附加组件时部署的版本可能并非最新可用版本。在之前的输出中，创建附加组件时已部署最新的版本。

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 v1.11.4-eksbuild.1 \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --resolve-
conflicts PRESERVE
```

PRESERVE（保留）选项将保留您为附加组件设置的任何自定义设置。有关此设置的其他选项的更多信息，请参阅《Amazon EKS 命令行参考》中的更新附加组件。有关 Amazon EKS 附加组件配置管理的更多信息，请参阅 Kubernetes 字段管理。

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。此选项会将附加组件作为自行管理安装使用，而不是作为 Amazon EKS 附加组件使用。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`、AWS Management Console 或 AWS 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
```

AWS Management Console

要使用 AWS 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. 选择附加组件选项卡。
5. 选择 Amazon EBS CSI 驱动程序。
6. 选择移除。
7. 在删除：aws-ebs-csi-driver 确认对话框中，执行以下操作：
   a. 如果希望 Amazon EKS 停止管理附加组件设置，请选择在集群上保留。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。
   b. 输入 **aws-ebs-csi-driver**。
   c. 选择 Remove（移除）。

AWS CLI

要使用 AWS 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.git
```

2. 导航到 dynamic-provisioning 示例目录。

```
cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
```

3. （可选）默认情况下，manifests/storageclass.yaml 文件预置 gp2 Amazon EBS 卷。要改用 gp3 卷，请将 type: gp3 添加到 manifests/storageclass.yaml。

```
echo "parameters:
  type: gp3" >> manifests/storageclass.yaml
```

4. 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/
```

5. 描述 ebs-sc 存储类。

```
kubectl describe storageclass ebs-sc
```

示例输出如下。

```
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。

6. 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。

7. 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
```

示例输出如下。

```
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
```

8. 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
```

示例输出如下。

```
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。

9. 验证 Pod 是否将数据写入卷。

```
kubectl exec -it app -- cat /data/out.txt
```

示例输出如下。

```
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
[...]
```

10. 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

> **Important**
>
> 如果您在版本 1.22 或更早版本的集群上运行 Pods，则必须先安装 Amazon EBS CSI 驱动程序，然后再将集群更新到版本 1.23，以避免服务中断。

Amazon EBS 容器存储接口 (CSI) 迁移功能将处理存储操作的责任从 Amazon EBS 树内 EBS 存储预配置程序移交给 Amazon EBS CSI 驱动程序。

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。
- 提供简化的附加组件模型，使 AWS 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 Kubernetes1.23: 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。

> **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-ebs StorageClass 卷？

是的，只要安装了 Amazon EBS CSI 驱动程序即可执行此操作。

kubernetes.io/aws-ebs StorageClass 置备程序是否会从 Amazon EKS 中删除？

kubernetes.io/aws-ebs StorageClass 置备程序和 awsElasticBlockStore 卷类型不再受支持，但没有删除它们的计划。这些资源被视为 Kubernetes API 的一部分。

如何安装 Amazon EBS CSI 驱动程序？

我们建议安装 Amazon EBS CSI 驱动程序 Amazon EKS 插件。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。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。

> **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 Amazon EKS 集群 IAM 角色 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，请参阅创建 Amazon EBS CSI 驱动程序 IAM 角色。

如何检查我的集群中是否安装了 Amazon EBS CSI 驱动程序？

要确定集群上是否安装了驱动程序，请运行以下命令：

```
kubectl get csidriver ebs.csi.aws.com
```

要检查该安装是否由 Amazon EKS 管理，请运行以下命令：

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

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。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 AWS 博客上的将 Amazon EKS 集群从 gp2 迁移到 gp3 EBS 卷。

如何使用注释修改 Amazon EBS 卷？

从 aws-ebs-csi-driver v1.19.0-eksbuild.2 开始，您可以在其 PersistentVolumeClaim（PVC）内使用注释修改 Amazon EBS 卷。新的卷修改功能作为

附加的 sidecar 实施，名为 `volumemodifier`。有关更多信息，请参阅 AWS 博客上的 Simplifying Amazon EBS volume migration and modification on Kubernetes using the EBS CSI Driver。

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。

Amazon EFS CSI 驱动程序

Amazon Elastic File System（Amazon EFS）提供无服务器的完全弹性文件存储，因此，您无需预置或管理存储容量和性能，即可共享文件数据。Amazon EFS Container Storage Interface（CSI）驱动程序提供了一个 CSI 接口，允许在 AWS 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。

注意事项

- Amazon EFS CSI 驱动程序与基于 Windows 的容器映像不兼容。
- 不能将持久性卷的动态预置与 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 集群版本上通过 `1.1` 版本的驱动程序，对持久性卷使用静态预置。
- 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。
- 版本 `1.4.2` 或更高版本的此驱动程序支持使用 FIPS 装载文件系统。
- 注意 Amazon EFS 的资源配额。例如，可以为每个 Amazon EFS 文件系统创建 1000 个接入点的配额。有关更多信息，请参阅您无法更改的 Amazon EFS 资源配额。

先决条件

- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参

阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。

> **Note**
>
> AWS Fargate 上运行的 Pod 会自动挂载 Amazon EFS 文件系统。

创建 IAM 角色

Amazon EFS CSI 驱动程序需要 IAM 权限才能与您的文件系统进行交互。创建 IAM 角色并向其附加所需的 AWS 托管策略。您可以使用 `eksctl`、AWS Management Console或 AWS CLI。

> **Note**
>
> 此过程中的具体步骤是为将驱动程序用作 Amazon EKS 附加组件而编写的。有关自行管理的安装的详细信息，请参阅 GitHub 上的 Set up driver permission。

eksctl

使用 **`eksctl`** 创建 Amazon EFS CSI 驱动程序 IAM 角色

运行以下命令以创建 IAM 角色。将 *`my-cluster`* 替换为您的集群名称，并将 *`AmazonEKS_EFS_CSI_DriverRole`* 替换为您的角色名称。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
```

```
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

使用 AWS Management Console 创建 Amazon EFS CSI 驱动程序 IAM 角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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`。
   d. 选择下一步。
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 *AmazonEFSCSIDriverPolicy*。
   b. 选中搜索返回的 *AmazonEFSCSIDriverPolicy* 左侧的复选框。
   c. 选择下一步。
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_EFS_CSI_DriverRole***。
   b. 在添加标签（可选）下，将标签作为键值对附加，以将元数据添加到角色。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 资源。
   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/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

将以下行添加到上一行的上方。将 *region-code* 替换为集群所在的 AWS 区域。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为集群的 OIDC 提供商 ID。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:efs-csi-*",
```

10. 将 Condition 运算符从 "StringEquals" 修改为 "StringLike"。
11. 选择 Update policy（更新策略）以完成操作。

AWS CLI

使用 AWS CLI 创建 Amazon EFS CSI 驱动程序 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
```

示例输出如下。

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. 创建授予 AssumeRoleWithWebIdentity 操作权限的 IAM 角色。

   a. 将以下内容复制到名为 *aws-efs-csi-driver-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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 arn:aws: 替换为 arn:aws-us-gov:。

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

```
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:efs-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

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://"aws-efs-csi-driver-trust-
policy.json"
```

3. 使用以下命令以将所需的 AWS 托管策略附加到角色。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name AmazonEKS_EFS_CSI_DriverRole
```

安装 Amazon EFS CSI 驱动程序

我们建议您通过 Amazon EKS 附加组件方式来安装 Amazon EFS CSI 驱动程序。要将 Amazon EKS 附加组件添加到您的集群，请参阅 创建附加组件。有关附加组件的更多信息，请参阅 Amazon EKS 附

加组件。如果您无法使用 Amazon EKS 附加组件方式，我们鼓励您将有关您为什么无法使用的问题提交到 容器路线图 GitHub 存储库。

如果您想要自行管理 Amazon EFS CSI 驱动程序的安装，请参阅 GitHub 上的 安装。

创建 Amazon EFS 文件系统

要创建 Amazon EFS 文件系统，请参阅 GitHub 上的 为 Amazon EKS 创建 Amazon EKS 文件系统。

部署示例应用程序

您可以部署各种示例应用程序并根据需要对其进行修改。有关更多信息，请参阅 GitHub 上的 示例。

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。建议使用最新版本的驱动程序。有关可用版本，请参阅 GitHub 上的 CSI Specification Compatibility Matrix（CSI 规范兼容性矩阵）。

> **Note**
>
> Fargate 不支持该驱动程序。

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：

- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

- 您的设备或 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。
- 您的设备或 AWS 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 1.29，则可以将 kubectl 的 1.28、1.29 或 1.30 版本与之配合使用。要安装或升级 kubectl，请参阅 安装或更新 kubectl。

以下步骤可帮助您使用 FSx for Lustre CSI 驱动程序创建简单的测试集群，以便您了解其工作原理。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 AWS 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 AWS 托管策略附加到该服务账户。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 arn:aws: 替换为 arn:aws-us-gov:。

```
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
```

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 AWS 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。用 *release-X.XX* 替换所需的分支。不支持主分支，因为它可能包含与当前发布的稳定版本驱动程序不兼容的即将推出的功能。建议使用最新发布的版本。有关活动分支的列表，请参阅 GitHub 上的 aws-fsx-csi-driver。

> **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=release-X.XX"
```

示例输出如下。

```
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 AWS CLI 输出中再提供它，您可以执行以下操作以在 AWS Management Console 中查看它。

   a. 打开 AWS 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 AWS 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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 arn:aws: 替换为 arn:aws-us-gov:。

```
kubectl annotate serviceaccount -n kube-system fsx-csi-controller-sa \
  eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSFSxLustreCSIDriverFullAccess --
overwrite=true
```

   示例输出如下。

```
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 AWS Management Console 中的 Networking（联网）部分下或通过使用以下 AWS 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 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 AWS 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
```

示例输出如下。

```
storageclass.storage.k8s.io/fsx-sc created
```

6. 下载持久卷注册清单。

```
curl -O 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
```

示例输出如下。

```
persistentvolumeclaim/fsx-claim created
```

9. 确认已预配置文件系统。

```
kubectl describe pvc
```

示例输出如下。

```
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
```

示例输出如下。

```
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
```

示例输出如下。

```
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
```

示例输出如下。

```
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。

适用于 NetApp ONTAP 的 Amazon FSx CSI 驱动程序

NetApp's 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 集群管理适用于 NetApp ONTAP 的 Amazon FSx 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 与适用于 NetApp ONTAP 的 Amazon FSx 结合使用。

适用于 NetApp ONTAP 的 Amazon FSx 是一种存储服务，使您可以在云中启动和运行完全托管的 ONTAP 文件系统。ONTAP 是一种 NetApp's 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。FSx for ONTAP 提供本地 NetApp 文件系统的功能、性能和 API，具有完全托管式 AWS 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 FSx for OpenZFS CSI 驱动程序

Amazon FSx for OpenZFS 是一项完全托管式的文件存储服务，可让您轻松地将数据从本地 ZFS 或其他基于 Linux 的文件服务器移动到 AWS。您可以在不更改应用程序代码或数据管理方式的情况下

执行此操作。它基于开源 OpenZFS 文件系统提供高度可靠、可扩展、高效且功能丰富的文件存储。它将这些功能与完全托管式 AWS 服务的灵活性、可扩展性和简单性相结合。有关更多信息，请参阅《Amazon FSx for OpenZFS 用户指南》https://docs.aws.amazon.com/fsx/latest/OpenZFSGuide/what-is-fsx.html。

Amazon FSx for OpenZFS Container Storage Interface (CSI) 驱动程序提供了一个 CSI 接口，允许 Amazon EKS 集群管理 Amazon FSx for OpenZFS 卷的生命周期。要将 Amazon FSx for OpenZFS CSI 驱动程序部署到您的 Amazon EKS 集群，请参阅 GitHub 上的 `aws-fsx-openzfs-csi-driver`。

Amazon File Cache CSI 驱动程序

Amazon File Cache 是 AWS 上一个完全托管的高速缓存，用于处理文件数据，无论数据存储在何处。Amazon File Cache 会在首次访问数据时自动将数据加载到缓存中，并在不使用时释放数据。有关更多信息，请参阅《Amazon File Cache 用户指南》。

Amazon File Cache Storage Interface (CSI) 驱动程序提供了一个 CSI 接口，允许 Amazon EKS 集群管理 Amazon 文件缓存的生命周期。要将 Amazon File Cache CSI 驱动程序部署到您的 Amazon EKS 集群，请参阅 GitHub 上的 `aws-file-cache-csi-driver`。

适用于 Amazon S3 的 Mountpoint CSI 驱动程序

借助适用于 Amazon S3 的 Mountpoint 容器存储接口（CSI）驱动程序，您的 Kubernetes 应用程序可以通过文件系统接口访问 Amazon S3 对象，从而在不更改任何应用程序代码的情况下实现高聚合吞吐量。基于适用于 Amazon S3 的 Mountpoint 构建的 CSI 驱动程序，将 Amazon S3 存储桶以卷的形式呈现，可供 Amazon EKS 和自我管理的 Kubernetes 集群中的容器访问。本主题介绍了如何将适用于 Amazon S3 的 Mountpoint CSI 驱动程序部署到您的 Amazon EKS 集群。

注意事项

- 适用于 Amazon S3 的 Mountpoint CSI 驱动程序目前与基于 Windows 的容器映像不兼容。
- 适用于 Amazon S3 的 Mountpoint CSI 驱动程序不支持 AWS Fargate。但支持在 Amazon EC2 中运行的容器（无论是 Amazon EKS 还是自定义 Kubernetes 安装）。
- 适用于 Amazon S3 的 Mountpoint CSI 驱动程序仅支持静态预置。不支持动态预置或创建新存储桶。

> **Note**
>
> 静态预调配是指使用现有的 Amazon S3 存储桶，该存储桶被指定为 `PersistentVolume` 对象中 `volumeAttributes` 中的 `bucketName`。有关更多信息，请参阅 GitHub 上的静态预置。

- 使用适用于 Amazon S3 的 Mountpoint CSI 驱动程序挂载的卷，不支持所有 POSIX 文件系统功能。有关文件系统行为的详细信息，请参阅 GitHub 上的适用于 Amazon S3 的 Mountpoint 文件系统行为。

先决条件

- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- 您的设备或 AWS CloudShell 上已安装并配置 2.12.3 或更高版本的 AWS CLI。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。

创建 IAM 策略

适用于 Amazon S3 的 Mountpoint CSI 驱动程序需要 Amazon S3 权限才能与文件系统交互。本节将介绍如何创建 IAM 策略来授予必要的权限。

以下示例策略遵循 Mountpoint 的 IAM 权限建议。或者，您可以使用 AWS 托管策略 AmazonS3FullAccess，但此托管策略授予的权限超出了 Mountpoint 所需的权限。

有关 Mountpoint 建议权限的更多信息，请参阅 GitHub 上的 Mountpoint IAM 权限。

使用 IAM 控制台创建 IAM 策略

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 在策略页面上，选择创建策略。
4. 对于策略编辑器，选择 JSON。

5. 在策略编辑器下，复制并粘贴以下内容：

> **Important**
>
> 将 DOC-EXAMPLE-BUCKET1 替换为您自己的 Amazon S3 存储桶名称。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
            ]
        }
   ]
}
```

与 Amazon S3 Express One Zone 存储类一起推出的目录存储桶，使用不同于一般用途存储桶的身份验证机制。您应该使用 `s3express:CreateSession` 操作，而非使用 `s3:*` 操作。有关目录存储桶的更多信息，请参阅《Amazon S3 用户指南》中的目录存储桶。

以下是您将对目录存储桶使用的最低权限策略的示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:aws-region:111122223333:bucket/DOC-
EXAMPLE-BUCKET1--az_id--x-s3"
        }
    ]
}
```

6. 选择下一步。
7. 在查看和创建页面上，为您的策略命名。此示例演练使用名称 AmazonS3CSIDriverPolicy。
8. 选择创建策略。

创建 IAM 角色

适用于 Amazon S3 的 Mountpoint CSI 驱动程序需要 Amazon S3 权限才能与文件系统交互。本节将介绍如何创建 IAM 角色来委派这些权限。要创建此角色，您可以使用 eksctl、IAM 控制台或 AWS CLI。

> **Note**
>
> 在上一节中创建了 IAM policy AmazonS3CSIDriverPolicy。

eksctl

要使用 **eksctl** 创建适用于 Amazon S3 的 Mountpoint CSI 驱动程序 IAM 角色

要创建 IAM 角色和 Kubernetes 服务账户，请运行以下命令。这些命令还会将 AmazonS3CSIDriverPolicy IAM 策略附加到角色，使用 IAM 角色的 Amazon 资源名称（ARN）对 Kubernetes 服务账户（s3-csi-controller-sa）进行注释，并将 Kubernetes 服务账户名称添加到 IAM 角色的信任策略。

```
CLUSTER_NAME=my-cluster
REGION=region-code
```

```
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

IAM console

要使用 AWS Management Console 创建适用于 Amazon S3 的 Mountpoint CSI 驱动程序 IAM 角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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）。

      如果未显示 URL，请查看先决条件部分。
   c. 对于 Audience (受众)，请选择 `sts.amazonaws.com`。
   d. 选择下一步。
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 **`AmazonS3CSIDriverPolicy`**。

      > **Note**
      > 在上一节中创建了此策略。

   b. 选中搜索返回的 `AmazonS3CSIDriverPolicy` 结果左侧的复选框。
   c. 选择下一步。
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_S3_CSI_DriverRole**。

b. 在添加标签（可选）下，将标签作为键值对附加，以将元数据添加到角色。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 资源。

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/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

在上一行末尾添加逗号，然后在逗号后添加下一行。将 *region-code* 替换为集群所在的 AWS 区域。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为集群的 OIDC 提供商 ID。

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-*"
```

10. 将 Condition 运算符从 "StringEquals" 变更为 "StringLike"。
11. 选择 Update policy（更新策略）以完成操作。

AWS CLI

要使用 AWS CLI 创建适用于 Amazon S3 的 Mountpoint CSI 驱动程序 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
```

示例输出如下。

```
https://oidc.eks.region-code.amazonaws.com/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 *aws-s3-csi-driver-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/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:s3-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

   b. 创建角色。您可以将 *AmazonEKS_S3_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_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. 使用以下命令将之前创建的 IAM 策略附加到角色。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> 在上一节中创建了 IAM 策略 AmazonS3CSIDriverPolicy。

4. 如果您将驱动程序作为 Amazon EKS 附加组件安装，则跳过此步骤。对于自行管理的驱动程序安装，请创建 Kubernetes 服务账户并使用您创建的 IAM 角色的 ARN 进行注释。

   a. 将以下内容保存到名为 *mountpoint-s3-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-mountpoint-s3-csi-driver
     name: mountpoint-s3-csi-controller-sa
     namespace: kube-system
     annotations:
       eks.amazonaws.com/role-arn:
    arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
   ```

   b. 在集群上创建 Kubernetes 服务账户。使用您创建的名为 *AmazonEKS_S3_CSI_DriverRole* 的 IAM 角色对 Kubernetes 服务账户（mountpoint-s3-csi-controller-sa）进行注释。

   ```
   kubectl apply -f mountpoint-s3-service-account.yaml
   ```

   > **Note**
   >
   > 在此过程中部署插件后，系统会创建一个名为 s3-csi-driver-sa 的服务账户，并将其配置为使用该服务账户。

安装适用于 Amazon S3 的 Mountpoint CSI 驱动程序

您可以通过 Amazon EKS 插件，安装适用于 Amazon S3 的 Mountpoint CSI 驱动程序。您可以使用 `eksctl`、AWS Management Console 或 AWS CLI 将附加组件添加到您的集群。

您可以选择将 Amazon Mountpoint S3 CSI 驱动程序作为自托管安装进行安装。有关执行自我管理安装的说明，请参阅 GitHub 上的安装。

eksctl

要使用 **`eksctl`** 添加 Amazon S3 CSI 附加组件

运行以下命令。将 *`my-cluster`* 替换为您的集群的名称，将 *`111122223333`* 替换为您的账户 ID，并将 *`AmazonEKS_S3_CSI_DriverRole`* 替换为之前创建的 IAM 角色。

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster --
service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
 --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。有关此设置其他选项的更多信息，请参阅 `eksctl` 文档中的 Addons（附加组件）。有关 Amazon EKS Kubernetes 字段管理的更多信息，请参阅 Kubernetes 字段管理。

AWS Management Console

要使用 AWS Management Console 添加适用于 Amazon EBS CSI 的 Mountpoint 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. 选择要为其配置适用于 Amazon S3 CSI 的 Mountpoint 附加组件的集群名称。
4. 选择附加组件选项卡。
5. 选择获取更多附加组件。
6. 在选择附加组件页面上，执行以下操作：
   a. 在 Amazon EKS 附加组件部分，选择适用于 Amazon S3 CSI 驱动程序的 Mountpoint复选框。
   b. 选择下一步。

7. 在配置选定的附加组件设置页面上，执行以下操作：

   a. 选择您想使用的 Version（版本）。

   b. 对于选择 IAM 角色，选择要将适用于 Amazon EBS CSI 驱动程序的 Mountpoint IAM 策略附加到的 IAM 角色的名称。

   c. （可选）展开可选配置设置。如果对冲突解决方法选择覆盖，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。

   d. 选择下一步。

8. 在查看和添加页面上，选择创建。附加组件安装完成后，您将看到已安装的附加组件。

AWS CLI

要使用 AWS CLI 添加适用于 Amazon EBS CSI 的 Mountpoint 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_S3_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-mountpoint-s3-csi-driver \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

配置适用于 Amazon S3 的 Mountpoint

在大多数情况下，您可以仅使用存储桶名称来配置适用于 Amazon S3 的 Mountpoint。有关配置适用于 Amazon S3 的 Mountpoint 的说明，请参阅 GitHub 上的配置适用于 Amazon S3 的 Mountpoint。

部署示例应用程序

您可以将静态预置部署到现有 Amazon S3 存储桶上的驱动程序。有关更多信息，请参阅 GitHub 上的静态预置。

删除适用于 Amazon S3 CSI 驱动程序的 Mountpoint

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。此选项会将附加组件作为自行管理安装使用，而不是作为 Amazon EKS 附加组件使用。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`、AWS Management Console 或 AWS CLI 移除 Amazon S3 CSI 附加组件。

eksctl

要使用 **`eksctl`** 移除 Amazon S3 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-mountpoint-s3-csi-driver --
preserve
```

AWS Management Console

要使用 AWS Management Console 移除 Amazon S3 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. 选择附加组件选项卡。
5. 选择适用于 Amazon S3 CSI 驱动程序的 Mountpoint。
6. 选择移除。
7. 在删除：aws-mountpoint-s3-csi-driver 确认对话框中，执行以下操作：
   a. 如果希望 Amazon EKS 停止管理附加组件设置，请选择在集群上保留。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。
   b. 输入 **`aws-mountpoint-s3-csi-driver`**。
   c. 选择移除。

AWS CLI

要使用 AWS CLI 移除 Amazon S3 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-mountpoint-s3-csi-driver --preserve
```

CSI 快照控制器

容器存储接口（CSI）快照控制器允许在兼容的 CSI 驱动程序（例如 Amazon EBS CSI 驱动程序）中使用快照功能。

以下是使用 CSI 快照控制器时需要考虑的一些事项。

- 快照控制器必须与具有快照功能的 CSI 驱动程序一起安装。Amazon EBS CSI 驱动程序支持创建 Amazon EBS CSI 托管卷的 Amazon EBS 快照。有关安装说明，请参阅Amazon EBS CSI 驱动程序。
- Kubernetes 不支持通过 CSI 迁移提供的卷的快照，例如使用 `StorageClass` 和预置器 `kubernetes.io/aws-ebs` 的 Amazon EBS 卷。创建卷时必须使用引用 CSI 驱动程序预置器 `ebs.csi.aws.com` 的 `StorageClass` 创建卷。有关 CSI 迁移的更多信息，请参阅 Amazon EBS CSI 迁移常见问题。

我们建议通过 Amazon EKS 托管的插件安装 CSI 快照控制器。要将 Amazon EKS 附加组件添加到您的集群，请参阅 创建附加组件。有关附加组件的更多信息，请参阅 Amazon EKS 附加组件。

或者，如果您想自我管理 Amazon EBS CSI 快照控制器的安装，请参阅 GitHub 上的上游 Kubernetes `external-snapshotter` 中的使用情况。

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

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 AWS CloudFormation 模板创建一个。如果您要在 AWS Outposts 上创建本地或扩展集群，请参阅 Amazon EKS 本地集群 VPC 及子网的要求和注意事项 而不是本主题。

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。

  为此，您可以更新集群以更改该集群使用的子网和安全组。您可以从 AWS Management Console、AWS CLI 的最新版本、AWS CloudFormation 以及 `eksctl` 版本 `v0.164.0-rc.0` 或更高版本进行更新。您可能需要执行此操作，为子网提供更多可用 IP 地址，以便成功升级集群版本。

  > **Important**
  >
  > 您添加的所有子网均必须位于与创建集群时最初提供的相同一组可用区中。新子网必须满足所有其他要求，例如，它们必须有足够的 IP 地址。

例如，假设您创建一个集群并指定四个子网。按照您指定的顺序，第一个子网位于 us-west-2a 可用区中，第二个和第三个子网位于 us-west-2b 可用区中，第四个子网位于 us-west-2c 可用区中。如果要更改子网，则必须在三个可用区中各提供至少一个子网，并且这些子网必须与原始子网位于同一个 VPC 中。

如果您需要的 IP 地址多于 VPC 中的 CIDR 块，则可以通过将其他无类别域间路由（CIDR）块与您的 VPC 关联来添加其他 CIDR 块。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 AWS PrivateLink 的 VPC 节点。有关更多信息，请参阅 子网要求和注意事项。

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：

键	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，您可以更改 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，或者在创建集群后更新子网。

集群的子网要求

您在创建或更新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 AWS Outposts、AWS Wavelength 或 AWS 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和 Kubernetes 资源。
- 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。
- 子网不能位于以下可用区中：

AWS 区域	区域名称	不允许使用的可用区 ID
us-east-1	美国东部（弗吉尼亚北部）	use1-az3
us-west-1	美国西部（加利福尼亚北部）	usw1-az2
ca-central-1	加拿大（中部）	cac1-az3

按组件划分的 IP 地址系列使用情况

下表包含 Amazon EKS 的每个组件所使用的 IP 地址系列。您可以使用网络地址转换（NAT）或其他兼容性系统从具有表条目 "No" 值的系列中的源 IP 地址连接到这些组件。

根据集群的 IP family（`ipFamily`）设置，功能可能会有所不同。此设置会更改 Kubernetes 分配给 Services 的 CIDR 数据块所使用的 IP 地址类型。设置值为 IPv4 的集群被称为 IPv4 cluster，设置值为 IPv6 的集群被称为 IPv6 cluster。

组件	仅限 **IPv4** 地址	仅限 **IPv6** 地址	双堆栈地址
EKS API 公用端点	是	否	否
EKS API VPC 端点	是	否	否
EKS Auth API 公用端点	是 [1]	是 [1]	是 [1]
EKS Auth API VPC 端点	是 [1]	是 [1]	是 [1]
EKS 集群公用端点	是	否	否
EKS 集群私有端点	是 [2]	是 [2]	不支持
EKS 集群子网	是 [2]	否	是 [2]
节点主 IP 地址	是 [2]	否	是 [2]
Service IP 地址的集群 CIDR 范围	是 [2]	是 [2]	不支持
来自 VPC CNI 的 Pod IP 地址	是 [2]	是 [2]	不支持

Note

[1] 端点是双堆栈，同时包含 IPv4 和 IPv6 地址。AWS 外部的应用程序、集群的节点以及集群内的容器组（pod）可以通过 IPv4 或 IPv6 到达此端点。

[2] 当您创建集群时，您可以在集群的 IP family（`ipFamily`）设置中在 IPv4 集群和 IPv6 集群之间进行选择，这是无法更改的。相反，在创建另一个集群并迁移工作负载时，您必须选择不同的设置。

节点的子网要求

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。
- 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，则该子网必须自动分配 IPv4 公有地址或 IPv6 地址。如果您将节点部署到具有关联的 IPv6 CIDR 块的私有子网，则该私有子网还必须自动分配 IPv6 地址。如果您使用 Amazon EKS AWS CloudFormation 模板在 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 AWS PrivateLink 的 VPC 端点到您的 VPC。您的节点和 Pods 需要与之通信的所有 AWS 服务 都需要 VPC 端点。示例包括 Amazon ECR、Elastic Load Balancing、Amazon CloudWatch、AWS 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 AWS 服务 都支持 VPC 端点。有关更多信息，请参阅什么是 AWS PrivateLink？和与 AWS PrivateLink 集成的 AWS 服务。有关更多 Amazon EKS 要求的列表，请参阅 私有集群要求。
- 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

键	值
kubernetes.io/role/internal-elb	1

- 公有子网

键	值
kubernetes.io/role/elb	1

当创建版本为 1.18 和更低版本的 Kubernetes 集群时，Amazon EKS 会将以下标签添加到指定的所有子网。

键	值
kubernetes.io/cluster/ *my-cluster*	shared

当您创建新 Kubernetes 集群时，Amazon EKS 不会将该标签添加到您的子网中。如果标签位于以前版本低于 1.19 的集群使用的子网上，则当该集群更新为较新版本时，不会自动从子网上移除该标签。版本 2.1.1 或更低版本的 AWS Load Balancer Controller 需要此标签。如果您使用的是较新版本的负载均衡器控制器，则可以删除该标签，而不会中断您的服务。

如果您使用 eksctl 或者任一 Amazon EKS AWS 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。
- 2020 年 3 月 26 之前 – 公有子网不会将公有 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 – 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、Windows 或 Arm 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。

共享子网要求和注意事项

您可以使用 VPC 共享与同一 AWS Organizations 中的其他 AWS 账户共享子网。您可以在共享子网中创建 Amazon EKS 集群，但要注意以下几点：

- VPC 子网的拥有者必须与参与者账户共享一个子网，该账户才能在其中创建 Amazon EKS 集群。
- 您不能使用 VPC 的默认安全组启动资源，因为此安全组属于拥有者。此外，参与者无法使用其他参与者或拥有者拥有的安全组启动资源。
- 在共享子网中，参与者和拥有者分别控制各自账户中的安全组。子网拥有者可以看到参与者创建的安全组，但不能对其执行任何操作。如果子网拥有者想要删除或修改安全组，则创建安全组的参与者必须执行该操作。
- 如果集群是由参与者创建的，则要注意以下几点：
  - 必须在该账户中创建集群 IAM 角色和节点 IAM 角色。有关更多信息，请参阅Amazon EKS 集群 IAM 角色 和Amazon EKS 节点 IAM 角色。
  - 所有节点必须由同一个参与者创建，包括托管节点组。
- 共享 VPC 拥有者无法查看、更新或删除参与者在共享子网中创建的集群。这是对每个账户具有不同访问权限的 VPC 资源的补充。有关更多信息，请参阅 Amazon VPC 用户指南中的拥有者和参与者的责任和权限。
- 如果您使用 Amazon VPC CNI plugin for Kubernetes 的自定义网络功能，则需要使用拥有者账户中列出的可用区 ID 映射来创建每个 `ENIConfig` 可用区。有关更多信息，请参阅 容器组（pod）的自定义网络。

有关 VPC 子网共享的更多信息，请参阅 Amazon VPC 用户指南中的与其他账户共享 VPC。

为 Amazon EKS 集群创建 VPC

您可以使用 Amazon Virtual Private Cloud（Amazon VPC）将AWS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 Amazon Web Services 的可扩展基础设施的优势。我们建议，在部署生产 Amazon EKS 集群之前，深入了解 Amazon VPC 服务。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 Amazon EKS VPC 和子网要求和注意事项 中所述的要求。本主题介绍如何使用 Amazon EKS 提供的 AWS CloudFormation 模板创建符合 Amazon EKS 要求的 VPC。部署模板后，您可以查看该模板创建的资源，以确切了解它创建了哪些资源以及这些资源的配置。

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 AWS 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 AWS 服务 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 AWS 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 AWS 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 之一粘贴到 Amazon S3 URL 下方的文本区域，然后选择 Next（下一步）：

   - IPv4

     ```
     https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
     ```

   - IPv4 和 IPv6

     ```
     https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-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 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - VpcBlock：为您的 VPC 选择 IPv4 CIDR 范围。您部署的每个节点、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，系统将从 Amazon 的全局单播地址空间自动为您分配 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，将在模板中为您指定此块。
   - 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，将在模板中为您指定此块。
   - 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，将在模板中为您指定此块。
   - 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，将在模板中为您指定此块。

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。
12. 如果您创建了 IPv4 VPC，请跳过此步骤。如果您创建了 IPv6 VPC，则必须为模板创建的公有子网启用自动分配 IPv6 地址选项。私有子网的该设置已启用。要启用该设置，请完成以下步骤：

    a. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
    b. 请在左侧导航窗格中，选择 Subnets（子网）
    c. 请选择其中一个公有子网（***stack-name***/SubnetPublic01 或 ***stack-name***/SubnetPublic02，包含 public 字样），然后依次选择 Actions（操作）、Edit subnet settings（编辑子网设置）。
    d. 选择 Enable auto-assign **IPv6** address（启用自动分配地址）复选框，然后选择 Save（保存）。
    e. 为另一个公有子网再次完成上述步骤。

Only public subnets

此 VPC 有三个部署到 AWS 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 AWS 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 AWS 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 Template is ready（模板已就绪）处于选中状态，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.us-west-2.amazonaws.com/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 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域和 AWS 账户，该名称必须在其内具有唯一性。
   - VpcBlock：为您的 VPC 选择 CIDR 块。您部署的每个节点、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。
12. （可选）您部署到此 VPC 的任何集群都能够将私有 IPv4 地址分配给您的 Pods 和 services。如果想要将集群部署到此 VPC 以将私有 IPv6 地址分配给您的 Pods 和 services，则必须对您的 VPC、子网、路由表和安全组进行更新。有关更多信息，请参阅《Amazon VPC 用户指南》中的将现有 VPC 从 IPv4 迁移到 IPv6。Amazon EKS 要求您的子网启用 Auto-assign IPv6 地址选项。该选项默认已禁用。

Only private subnets

此 VPC 有三个部署到 AWS 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 AWS PrivateLink 为节点通常需要访问的几个 AWS 服务 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 私有集群要求 以了解其他要求和考虑因素。

要创建您的 VPC

1. 打开 AWS 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 AWS 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 Template is ready（模板已就绪）处于选中状态，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.us-west-2.amazonaws.com/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 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - VpcBlock：为您的 VPC 选择 CIDR 块。您部署的每个节点、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。
12. （可选）您部署到此 VPC 的任何集群都能够将私有 IPv4 地址分配给您的 Pods 和 services。如果想要将集群部署到此 VPC 以将私有 IPv6 地址分配给您的 Pods 和 services，则必须对您的 VPC、子网、路由表和安全组进行更新。有关更多信息，请参阅《Amazon VPC 用户指南》中的将现有 VPC 从 IPv4 迁移到 IPv6。Amazon EKS 要求您的子网启用 Auto-assign IPv6 地址选项（该选项默认为禁用）。

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	来源	目标位置
入站	全部	全部	自身	
出站	全部	全部		0.0.0.0/0（IPv4）或 ::/0（IPv6）

> **Important**
>
> 如果您的集群不需要出站规则，则可以将其删除。如果将其删除，您仍须遵守限制集群流量中列出的最少规则。如果您删除入站规则，则每当集群更新时，Amazon EKS 都会重新创建该规则。

Amazon EKS 会向安全组添加以下标签。如果您删除标签，则每当集群更新时，Amazon EKS 都会将其重新向安全组添加标签。

键	值
kubernetes.io/cluster/ *my-cluster*	owned

键	值
aws:eks:cluster-name	*my-cluster*
Name	eks-cluster-sg- *my-cluste r* -*uniqueid*

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AWS 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 AWS 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，则可以删除默认出站规则并添加集群所需的以下最少规则。如果您删除默认入站规则，则每当集群更新时，Amazon EKS 都会重新创建该规则。

Rule type	协议	端口	目标位置
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 了解其他注意事项。
- 从 Amazon ECR 或要从中拉取映像的其他容器注册表 API（例如 DockerHub）中拉取容器映像的节点访问权限。有关更多信息，请参阅 AWS 一般参考 中的 AWS 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 AWS CLI 或 AWS 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 联网附加组件

有多个联网附加组件可用于 Amazon EKS 集群。

内置附加组件

> **Note**
>
> 如果集群是使用除控制台以外的任何方式创建的，则每个集群都将附带内置附加组件的自行管理版本。自行管理的版本不能从 AWS Management Console、AWS Command Line Interface 或 SDK 进行管理。您可以管理自行管理附加组件的配置和升级。
> 建议您向集群添加 Amazon EKS 类型的附加组件，而不是自行管理类型的附加组件。如果集群是通过控制台创建的，则会安装这些附加组件的 Amazon EKS 类型。

Amazon VPC CNI plugin for Kubernetes

此 CNI 附加组件会创建弹性网络接口并将其附加到您的 Amazon EC2 节点。附加组件还会将 VPC 中的私有 IPv4 或 IPv6 地址分配给每个 Pod 和服务。您的集群上默认安装此附加组件。有关更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。

CoreDNS

CoreDNS 是一个灵活、可扩展的 DNS 服务器，可用作 Kubernetes 集群 DNS。CoreDNS 为集群中的所有 Pods 提供名称解析服务。您的集群上默认安装此附加组件。有关更多信息，请参阅 使用 CoreDNS Amazon EKS 附加组件。

kube-proxy

此附加组件负责维护 Amazon EC2 节点上的网络规则，并实现与 Pods 的网络通信。您的集群上默认安装此附加组件。有关更多信息，请参阅 使用 Kubernetes kube-proxy 附加组件。

可选 AWS 联网附加组件

AWS Load Balancer Controller

当您部署 loadbalancer 类型的 Kubernetes 服务对象时，控制器会创建 AWS 网络负载均衡器。创建 Kubernetes 传入对象时，控制器会创建 AWS 应用程序负载均衡器。我们建议使用此控制器来预置网络负载均衡器，而不是使用 Kubernetes 内置的旧版云提供商控制器。有关更多信息，请参阅AWS Load Balancer Controller 文档。

AWS 网关 API 控制器

借助此控制器，您可以使用 Kubernetes 网关 API 跨多个 Kubernetes 集群连接服务。控制器使用 Amazon VPC Lattice 服务连接在 Amazon EC2 实例、容器和无服务器功能上运行的 Kubernetes 服务。有关更多信息，请参阅 AWS Gateway API 控制器文档。

有关附加组件的更多信息，请参阅 Amazon EKS 附加组件。

使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件

Amazon VPC CNI plugin for Kubernetes 附加组件部署在 Amazon EKS 集群中的每个 Amazon EC2 节点上。附加组件会创建弹性网络接口并将其附加到 Amazon EC2 节点。附加组件还会将 VPC 中的私有 IPv4 或 IPv6 地址分配给每个 Pod 和服务。

附加组件版本随集群中的每个 Fargate 节点一起部署，但您无需在 Fargate 节点上对其进行更新。其他兼容的 CNI 插件也可用于 Amazon EKS 集群，但这是 Amazon EKS 唯一支持的 CNI 插件。

下表列出了每个 Kubernetes 版本的 Amazon EKS 附加组件类型的最新可用版本。

Kubernetes 版本	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
Amazon EKS 类型的 VPC CNI 版本	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuilc	v1.18.2-eksbuild.1

Important

如果您自行管理此附加组件，则表中的版本可能与可用的自行管理版本不同。有关更新此附加组件的自行管理类型的更多信息，请参阅更新自我管理的附加组件。

Important

要升级到 VPC CNI v1.12.0 或更高版本，必须先升级到 VPC CNI v1.7.0。建议您一次更新一个次要版本。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。
- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- 附加有 AmazonEKS_CNI_Policy IAM 策略（如果集群使用 IPv4 系列）或 IPv6 策略（如果集群使用 IPv6 系列）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（IRSA）。
- 如果使用是版本 1.7.0 或更高版本的 Amazon VPC CNI plugin for Kubernetes，并且使用自定义 Pod 安全策略，请参阅 删除默认的 Amazon EKS Pod 安全策略容器组（pod）安全策略。

- > **Important**
  >
  > Amazon VPC CNI plugin for Kubernetes 版本 `v1.16.0` 到 `v1.16.1` 已删除与 Kubernetes 版本 `1.23` 以及更早版本的兼容性。VPC CNI 版本 `v1.16.2` 还原与 Kubernetes 版本 `1.23` 以及更早版本和 CNI 规范 `v0.4.0` 的兼容性。

  Amazon VPC CNI plugin for Kubernetes 版本 `v1.16.0` 到 `v1.16.1` 实现 CNI 规范版本 `v1.0.0`。运行 Kubernetes 版本 `v1.24` 或更高版本的 EKS 集群上支持 CNI 规范 `v1.0.0`。Kubernetes 版本 `v1.23` 或更早版本上不支持 VPC CNI 版本 `v1.16.0` 到 `v1.16.1`，也不支持 CNI 规范 `v1.0.0`。有关 CNI 规范的更多信息`v1.0.0`，请参阅容器网络接口 (CNI) 规范

注意事项

- 版本指定为 `major-version.minor-version.patch-version-eksbuild.build-number`。
- 检查每个功能的版本兼容性

  每个版本的某些功能都Amazon VPC CNI plugin for Kubernetes需要特定的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。除非您出于某个特定原因需要运行早期版本，否则建议您运行最新版本。

创建 Amazon EKS 附加组件

创建 Amazon EKS 类型的附加组件。

1. 查看集群上当前安装的附加组件版本。

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
    | cut -d : -f 3
   ```

   示例输出如下。

   ```
   v1.16.4-eksbuild.2
   ```

2. 查看集群上当前安装的附加组件类型。根据您创建集群时使用的工具，您的集群上目前可能没有安装 Amazon EKS 附加组件类型。将 *my-cluster* 替换为您集群的名称。

```
$ aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

如果返回来的是版本号，则表明您的集群上安装有 Amazon EKS 类型的附加组件，而且此流程中其余的步骤，您也不需要走完。如果返回来的是一个错误，则表明您的集群上没有安装 Amazon EKS 类型的附加组件。要安装，就需完成此流程中其余的步骤。

3. 保存您当前安装的附加组件的配置。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. 使用 AWS CLI 创建附加组件。如果要使用 AWS Management Console 或 `eksctl` 来创建附加组件，请参阅 创建附加组件 并指定 `vpc-cni` 为附加组件名称。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令。

   - 将 *my-cluster* 替换为您的集群名称。
   - 将 *v1.18.2-eksbuild.1* 替换为您的集群版本的最新版本表中列出的最新版本。
   - 将 *111122223333* 替换为您的账户 ID，并将 *AmazonEKSVPCCNIRole* 替换为您创建的现有 IAM 角色的名称。指定角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅 为集群创建 IAM OIDC 提供商。

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.18.2-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

如果您对当前附加组件应用的自定义设置与 Amazon EKS 附加组件的默认设置相冲突，则创建可能会失败。如果创建失败，您会收到一条可以帮助您解决问题的错误信息。或者，您可以将 **--resolve-conflicts OVERWRITE** 添加到前面的命令中。这样一来，附加组件会覆盖任何现有的自定义设置。创建附加组件后，您可以使用自定义设置对其进行更新。

5. 确认您的集群的 Kubernetes 版本的附加组件最新版本已添加到您的集群。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
```

附加组件创建可能需要几秒钟才能完成。

示例输出如下。

```
v1.18.2-eksbuild.1
```

6. 如果您在创建 Amazon EKS 附加组件之前对原始附加组件进行了自定义设置，则请使用您在上一步中保存的配置，以使用您的自定义设置更新 Amazon EKS 附加组件。
7. （可选）将 `cni-metrics-helper` 安装到您的集群。它会抓取弹性网络接口和 IP 地址信息，在集群级别聚合这些信息，并将指标发布到 Amazon CloudWatch。有关更多信息，请参阅 GitHub 上的 cni-metrics-helper。

更新 Amazon EKS 附加组件

更新 Amazon EKS 类型的附加组件。如果您尚未将 Amazon EKS 类型的附加组件添加到集群中，则请添加它或查看 更新自我管理的附加组件，而不是完成此过程。

1. 查看集群上当前安装的附加组件版本。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
```

示例输出如下。

```
v1.16.4-eksbuild.2
```

如果返回的版本与最新版本表中集群的 Kubernetes 版本的版本相同，则您的集群上已经安装了最新版本，且您无需完成此过程的其余部分。如果您在输出中收到错误信息而不是版本号，则您的集群上没有安装 Amazon EKS 类型的附加组件。您需要先创建附加组件，然后才能使用此过程对其进行更新。

2. 保存您当前安装的附加组件的配置。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. 使用 AWS CLI 更新您的附加组件。如果您想要使用 AWS Management Console 或 `eksctl` 更新附加组件，则请参阅 更新附加组件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令。

- 将 *my-cluster* 替换为您的集群名称。
- 将 *v1.18.2-eksbuild.1* 替换为您的集群版本的最新版本表中列出的最新版本。
- 将 *111122223333* 替换为您的账户 ID，并将 *AmazonEKSVPCCNIRole* 替换为您创建的现有 IAM 角色的名称。指定角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅 为集群创建 IAM OIDC 提供商。
- **--resolve-conflicts** *PRESERVE* 选项保留附加组件的现有配置值。如果您为附加组件设置设定了自定义值，但未使用此选项，则 Amazon EKS 会使用其默认值覆盖您的值。如果您使用此选项，那么我们建议您在更新生产集群上的附加组件之前，先测试非生产集群上所有更改的字段和值。如果您将该值改为 OVERWRITE，则所有设置都将更改为 Amazon EKS 的默认值。如果您为任何设置设定了自定义值，这些值可能会被 Amazon EKS 的默认值覆盖。如果您将该值改为 none，Amazon EKS 不会更改任何设置的值，但更新可能会失败。如果更新失败，您会收到一条帮助您解决冲突的错误消息。
- 如果您没有更新配置设置，则请从命令中移除 **--configuration-values '{*"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*}'**。如果您更新配置设置，则将 *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}* 替换为您想要设置的设置。在此示例中，AWS_VPC_K8S_CNI_EXTERNALSNAT 环境变量设置为 `true`。您指定的值必须对配置架构有效。如果您不知道配置架构，请运行 **aws eks describe-addon-configuration --addon-name vpc-cni --addon-version *v1.18.2-eksbuild.1***，以将 *v1.18.2-eksbuild.1* 替换为您要查看配置的插件的版本号。将在输出中返回架构。如果您有任何现有的自定义配置，想要将其全部删除，并将所有设置的值设置回 Amazon EKS 的默认值，请从命令中删除 *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*，这样就可以有空的 **{}**。有关每项设置的说明，请参阅 GitHub 上的 CNI 配置变量。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.18.2-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

可能需要几秒钟才能完成更新。

4. 确认附加组件版本已更新。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

可能需要几秒钟才能完成更新。

示例输出如下。

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.18.2-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn":
 "arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

更新自我管理的附加组件

> **Important**
>
> 建议您向集群添加 Amazon EKS 类型的附加组件，而不是自行管理类型的附加组件。如果不熟悉这些类型之间的区别，请参阅 the section called “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 the section called “创建附加组件”。如果您无法使用 Amazon EKS 附加组件，我们鼓励您向容器路线图 GitHub 存储库提交有关您为什么无法使用的问题。

1. 确认已在您的集群上安装 Amazon EKS 类型的附加组件。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

如果返回错误消息，则表明您的集群上安装有 Amazon EKS 类型的附加组件。要自行管理附加组件，请完成此过程中的剩余步骤以更新附加组件。如果返回版本号，则表明集群上安装有 Amazon EKS 类型的附加组件。要对其进行更新，请使用 更新附加组件 中的过程，而不是此过程。如果不熟悉这些附加组件类型之间的区别，请参阅 Amazon EKS 附加组件。

2. 查看集群上当前安装的容器映像版本。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

示例输出如下。

```
v1.16.4-eksbuild.2
```

输出可能不包含版本号。

3. 备份当前设置，以便在更新版本后可以配置相同设置。

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. 要查看可用版本并熟悉要更新到的版本中的更改，请参阅 GitHub 上的 releases。请注意，我们建议更新到最新可用版本表中列出的相同 `major.minor.patch` 版本，即使 GitHub 上提供了更新版本。表中列出的构建版本并未在 GitHub 上列出的自行管理版本中指定。通过完成以下选项之一中的任务来更新您的版本：

   - 如果您的附加组件没有任何自定义设置，请在 GitHub 上针对要更新到的版本运行 `To apply this release:` 标题下的命令。
   - 如果有自定义设置，请使用以下命令下载清单文件。将 *https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.2/config/master/aws-k8s-cni.yaml* 更改为您要更新到的 GitHub 上的版本的 URL。

     ```
     curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.2/config/
     master/aws-k8s-cni.yaml
     ```

如有必要，使用您在上一步中创建的备份中的自定义设置修改清单，然后将修改后的清单应用到集群。如果节点无法访问从中提取映像的私有 Amazon EKS Amazon ECR 存储库（参阅清单中以 `image:` 开头的行），则您必须下载映像，将其复制到自己的存储库，然后修改清单以从存储库中提取映像。有关更多信息，请参阅 将容器镜像从一个存储库复制到另一个存储库。

```
kubectl apply -f aws-k8s-cni.yaml
```

5. 确认新版本现已安装在集群上。

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

示例输出如下。

```
v1.18.2
```

6. （可选）将 `cni-metrics-helper` 安装到您的集群。它会抓取弹性网络接口和 IP 地址信息，在集群级别聚合这些信息，并将指标发布到 Amazon CloudWatch。有关更多信息，请参阅 GitHub 上的 cni-metrics-helper。

配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户（IRSA）

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 AWS Identity and Access Management（IAM）权限。如果您的集群使用 IPv4 系列，则权限将在 `AmazonEKS_CNI_Policy` AWS 托管策略中指定。 如果您的集群使用 IPv6 系列，则权限必须添加到您创建的 IAM 策略。您可以将该策略附加到 Amazon EKS 节点 IAM 角色或单独 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 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。
- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。

步骤 1：创建 Amazon VPC CNI plugin for Kubernetes IAM 角色

创建 IAM 角色

1. 确定您的集群的 IP 系列。

   ```
   aws eks describe-cluster --name my-cluster | grep ipFamily
   ```

   示例输出如下。

   ```
   "ipFamily": "ipv4"
   ```

   输出可能会返回 `ipv6`。

2. 创建 IAM 角色。您可以使用 `eksctl` 或 `kubectl` 和 AWS 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 策略附加到该角色。此命令创建并部署一个创建 IAM 角色的 AWS 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。

   - `IPv4`

     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
```

- IPv6

  将 *my-cluster* 替换为您自己的值。将 *111122223333* 替换为您的账户 ID，并将 *AmazonEKS_CNI_IPv6_Policy* 替换为您的 IPv6 策略名称。如果没有 IPv6 策略，请参阅 为使用 IPv6 系列的集群创建 IAM 策略 创建一个。要将 IPv6 用于您的集群，它必须满足多项要求。有关更多信息，请参阅 集群、Pods 和 services 的 IPv6 地址。

  ```
  eksctl create iamserviceaccount \
      --name aws-node \
      --namespace kube-system \
      --cluster my-cluster \
      --role-name AmazonEKSVPCCNIRole \
      --attach-policy-arn
   arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
      --override-existing-serviceaccounts \
      --approve
  ```

kubectl and the AWS 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
   ```

   示例输出如下。

   ```
   https://oidc.eks.region-code.amazonaws.com/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。

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 AWS 区域。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
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 策略附加到角色。运行与集群的 IP 系列匹配的命令。

   - IPv4

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --role-name AmazonEKSVPCCNIRole
     ```

   - IPv6

将 *111122223333* 替换为您的账户 ID，并将 *AmazonEKS_CNI_IPv6_Policy* 替换为您的 IPv6 策略名称。如果没有 IPv6 策略，请参阅 为使用 IPv6 系列的集群创建 IAM 策略 创建一个。要将 IPv6 用于您的集群，它必须满足多项要求。有关更多信息，请参阅 集群、Pods 和 services 的 IPv6 地址。

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_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
```

3. （可选）配置您的 Kubernetes 服务账户使用的 AWS Security Token Service 端点类型。有关更多信息，请参阅 为服务账户配置 AWS Security Token Service 端点。

步骤 2：重新部署 Amazon VPC CNI plugin for KubernetesPods

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 *cpjw7* 替换为上一步输出中返回的其中一个 Pods 的名称。

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

示例输出如下。

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN:
 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:           /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

由于 Pod 包含两个容器，因此会返回两组重复的结果。两个容器具有相同的值。

如果您的 Pod 正在使用AWS 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 角色当前附加有 AmazonEKS_CNI_Policy IAM（IPv4）policy 或 IPv6 策略，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 AWS 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
```

- IPv6

  将 *111122223333* 替换为您的账户 ID，并将 *AmazonEKS_CNI_IPv6_Policy* 替换为您的 IPv6 策略名称。

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

为使用 **IPv6** 系列的集群创建 IAM 策略

如果您创建了使用 IPv6 系列的集群，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 策略，您可以将其分配给 IAM 角色。如果您现

有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 集群、Pods 和 services 的 IPv6 地址。

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 策略。

```
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 plugin for Kubernetes 为 Pods 提供联网功能。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 plugin for Kubernetes 设置。表中的所有信息仅适用于 Linux IPv4 节点。

Amazon EKS 节点类型	Amazon EC2			Fargate
使用案例	分配给网络接口的单个 IP 地址	分配给网络接口的 IP 前缀	Pods 的安全组	
容器组（pod）的自定义网络：分配来自不同于节点子网的子网的 IP 地址	支持	是	支持	是（通过 Fargate 配置文件控制的子网）
适用于 Pods 的 SNAT	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	容器组（pod）（如果您设置了 `POD_SECURITY_GROUP_ENFORCING_MODE =standard` 和 `AWS_VPC_K8S_CNI_EXTERNALSNAT =false`，则发往 VPC 外部端点的流量使用节点的安全组，而不是 Pod's 安全组）	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

Amazon EKS 节点类型	Amazon EC2			Fargate
使用案例	分配给网络接口的单个 IP 地址	分配给网络接口的 IP 前缀	Pods 的安全组	
网络策略（VPC CNI）	兼容	兼容	兼容 仅适用于 1.14.0 版本或更高版本的 Amazon VPC CNI 插件	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`	支持	不适用	不适用	不适用
`WARM_IP_TARGET`	支持	支持	不适用	不适用
`MINIMUM_IP_TARGET`	支持	支持	不适用	不适用
`WARM_PREFIX_TARGET`	不适用	支持	不适用	不适用

Note

- 不能将 IPv6 用于自定义联网。
- IPv6 地址不会进行转换，因此 SNAT 不适用。

- 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。
- 如果您使用 Calico 网络策略实施，则建议您将环境变量 `ANNOTATE_POD_IP` 设置为 `true`，以避免出现 Kubernetes 的已知问题。要使用此功能，必须将容器组（pod）的 `patch` 权限添加到 `aws-node` ClusterRole。请注意，向 `aws-node` DaemonSet 添加补丁权限会提高插件的安全范围。有关更多信息，请参阅 GitHub 上的 VPC CNI 存储库中的 ANNOTATE_POD_IP。
- 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

每个节点仅支持一个网络接口。您可以使用辅助 IPv4 地址和 IPv4 前缀。默认情况下，节点上可用的 IPv4 地址数等于可分配给每个弹性网络接口的辅助 IPv4 地址数减 1。但是，您可以通过启用 IP 前缀来增加节点上可用的 IPv4 地址和 Pod 密度。有关更多信息，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量。

Windows 支持 Calico 网络策略。您不能在 Windows 上使用 Pods 的安全组或自定义网络。

集群、Pods 和 services 的 **IPv6** 地址

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。
- 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，会部署此版本或更高版本。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。
- 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 AWS 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 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 AWS Outposts 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。
- 您可以使用 2.3.1 版或更高版本的 AWS 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 或到 IPv6 Pods 的网络流量进行负载均衡。有关更多信息，请参阅 AWS Load Balancer Controller 是什么？。

- 您必须将 IPv6 IAM 策略附加到节点 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 策略 和 步骤 1：创建 Amazon VPC CNI plugin for Kubernetes IAM 角色。
- 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 AWS 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。
- 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 AWS 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、托管节点组 和本主题中的注意事项。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 AWS 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 1.29，则可以将

kubectl 的 1.28、1.29 或 1.30 版本与之配合使用。要安装或升级 kubectl，请参阅 安装或更新 kubectl。

- 您正在使用的 IAM 安全主体必须具有使用 Amazon EKS IAM 角色、服务相关角色、AWS CloudFormation、VPC 和相关资源的权限。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 AWS CLI 创建资源的程序。您还可以使用 AWS 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 0.183.0 或更高版本。要安装或更新该工具，请参阅 eksctl 文档中的安装。

使用 **eksctl** 部署 **IPv6** 集群

1. 创建 ipv6-cluster.yaml 文件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令：

   - 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - 将 *region-code* 替换为 Amazon EKS 支持的任一 AWS 区域。有关 AWS 区域列表，请参阅 AWS 一般参考中的 Amazon EKS 端点和配额。
   - version 的值与您的集群版本有关。有关更多信息，请参阅支持的 Amazon EKS Kubernetes 版本。
   - 将 *my-nodegroup* 替换为您的节点组名称。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。
   - 将 *t3.medium* 替换为任何 AWS Nitro 系统实例类型。

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
```

```
  name: my-cluster
  region: region-code
  version: "X.XX"

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
```

示例输出如下。

```
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
```

示例输出如下。

```
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）部署示例应用程序或部署 AWS Load Balancer Controller 和示例应用程序以对应用程序或至 IPv6 Pods 的网络流量进行负载均衡。

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
```

AWS CLI

先决条件

在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。如果您使用 AWS CloudShell，则可能需要安装版本 2.12.3 或更高版本或者 1.27.160 版或更高版本的 AWS CLI，因为 AWS CloudShell 中安装的默认 AWS 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。要查看当前用户，请运行以下命令：

  ```
  aws sts get-caller-identity
  ```

- 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 AWS 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 AWS CLI 创建集群

将此程序步骤中的所有 *example values* 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 AWS 区域。该值可以是 Amazon EKS 支持的任何 AWS 区域。有关 AWS 区域列表，请参阅 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。将 *my-nodegroup* 替换为您的节点组名称。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。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 AWS 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 AWS CloudFormation 模板创建 IPv6 VPC。

   ```
   aws cloudformation create-stack --region $region_code --stack-name
    $vpc_stack_name \
     --template-url https://s3.us-west-2.amazonaws.com/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
   ```

   示例输出如下。

   ```
   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 AWS 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 AWS 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 策略附加到角色。

```
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。

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --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。

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 策略。

```
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 策略附加到 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（IRSA）。

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
```

示例输出如下。

```
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 AWS 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
```

示例输出如下。

```
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
```

示例输出如下。

```
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）部署示例应用程序或部署 AWS Load Balancer Controller 和示例应用程序以对应用程序或至 IPv6 Pods 的网络流量进行负载均衡。
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 AWS 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 集群、Pods 和 services 的 IPv6 地址。

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，Amazon VPC CNI 插件（适用于 Linux 或 Windows）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 *。

> **Note**
>
> 对于 Windows 节点，还有其他详细信息需要考虑。默认情况下，适用于 Windows 的 VPC CNI 插件使用网络配置进行定义，该配置下 SNAT 不包括同一 VPC 内向目标发送的流量。这意味着内部 VPC 通信已禁用 SNAT，分配给 Pod 的 IP 地址可在 VPC 内路由。但是向 VPC 以外目标发送的流量会将源 Pod IP 经 SNAT 处理到实例 ENI 的主 IP 地址。适用于 Windows 的此默认配置可确保容器组（pod）能像主机实例一样访问您的 VPC 之外的网络。

由于此行为：

- 只有当运行互联网资源的节点分配了公共或弹性 IP 地址并且位于公共子网中时，您的 Pods 才能与互联网资源通信。公有子网的关联路由表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。
- 对于 `1.8.0` 之前的插件版本，如果网络或 VPC 中的资源使用 VPC 对等、中转 VPC 或 AWS Direct Connect 连接到您的集群 VPC，则无法在辅助弹性网络接口后启动与 Pods 的通信。但是您的 Pods 可以启动与这些资源的通信并接收它们的响应。

如果以下任一陈述在您的环境中成立，则使用以下命令更改默认配置。

- 您在网络或 VPC 中有资源使用 VPC 对等、中转 VPC 或 AWS Direct Connect 连接到您的集群 VPC，它们需要使用 `IPv4` 地址启动与 Pods 的通信，并且您的插件版本早于 `1.8.0`。
- 您的 Pods 位于私有子网中，需要向互联网进行出站通信。子网具有通往 NAT 网关的路由。

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> `AWS_VPC_K8S_CNI_EXTERNALSNAT` 和 `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` CNI 配置变量不适用于 Windows 节点。Windows 不支持禁用 SNAT。至于从 SNAT 中排除 `IPv4` CIDR 列表，您可以通过在 Windows 引导脚本中指定 `ExcludedSnatCIDRs` 参数来定义。有关使用该参数的更多信息，请参阅 引导脚本配置参数。

*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。

为 Kubernetes 网络策略配置集群

默认情况下，Kubernetes 对集群中任何 Pods 之间或 Pods 与任何其他网络中的资源之间的 IP 地址、端口或连接没有限制。您可以使用 Kubernetes 网络策略来限制进出 Pods 的网络流量。有关更多信息，请参阅 Kubernetes 文档中的网络策略。

如果您的集群上有 `1.13` 版本或更早版本的 Amazon VPC CNI plugin for Kubernetes，则需要实施第三方解决方案来将 Kubernetes 网络策略应用于集群。`1.14` 版本或更高版本的插件可以实施网络策略，因此您无需使用第三方解决方案。在本主题中，您将了解如何将集群配置为在不使用第三方附加组件的情况下在集群上使用 Kubernetes 网络策略。

Amazon VPC CNI plugin for Kubernetes 中的网络策略在以下配置中受支持。

- `1.25` 版本及更高版本的 Amazon EKS 集群。
- 集群上 1.14 版本或更高版本的 Amazon VPC CNI plugin for Kubernetes。
- 为 `IPv4` 或 `IPv6` 地址配置的集群。
- 您可以将网络策略与 Pods 的安全组结合使用。通过网络策略，您可以控制集群内的所有通信。借助 Pods 的安全组，您可以控制从 Pod 中的应用程序对 AWS 服务 的访问。
- 您可以将网络策略与自定义网络和前缀委派结合使用。

注意事项

- 当使用 Amazon VPC CNI plugin for Kubernetes 将 Amazon VPC CNI plugin for Kubernetes 网络策略应用于集群时，您只能将策略应用于 Amazon EC2 Linux 节点。无法将策略应用于 Fargate 或 Windows 节点。
- 如果您的集群当前使用第三方解决方案来管理 Kubernetes 网络策略，则可以将这些策略与 Amazon VPC CNI plugin for Kubernetes 结合使用。但是，您必须删除现有的解决方案，使其不再管理相同的策略。
- 您可以对同一 Pod 应用多个网络策略。当配置两个或多个选择相同 Pod 的策略时，所有策略都将应用于 Pod。
- 网络策略中每个 `ingress:` 或 `egress:` 选择器中每个协议的唯一端口组合数最多为 24 个。
- 对于任何 Kubernetes 服务，服务端口必须与容器端口相同。如果您使用的是命名端口，也请在服务规范中使用相同的名称。

- Pod 启动时的策略实施

  Amazon VPC CNI plugin for Kubernetes 为容器组（pod）配置网络策略，同时进行容器组（pod）预置。在为新的容器组（pod）配置所有策略之前，新的容器组（pod）中的容器将从默认允许策略开始。这称为标准模式。默认允许策略意味着允许所有进出新容器组（pod）的入口和出口流量。

  您可以通过在 VPC CNI `DaemonSet` 的 `aws-node` 容器中将环境变量 `NETWORK_POLICY_ENFORCING_MODE` 设置为 `strict` 来更改此默认网络策略。

  ```
  env:
    - name: NETWORK_POLICY_ENFORCING_MODE
      value: "strict"
  ```

  将 `NETWORK_POLICY_ENFORCING_MODE` 变量设置为 `strict` 后，使用 VPC CNI 的容器组（pod）从默认拒绝策略开始，然后将配置策略。这称为严格模式。在严格模式下，您必须为集群中容器组（pod）需要访问的每个端点制定网络策略。请注意，此要求适用于 CoreDNS 容器组（pod）。未为带有主机网络的容器组（pod）配置默认拒绝策略。

- 网络策略功能创建并需要一个名为 `policyendpoints.networking.k8s.aws` 的 `PolicyEndpoint` 自定义资源定义（CRD）。自定义资源的 `PolicyEndpoint` 对象由 Amazon EKS 管理。您不应该修改或删除这些资源。
- 如果您运行的容器组（pod）使用实例角色 IAM 凭证或连接到 EC2 IMDS，请注意检查是否存在会阻止访问 EC2 IMDS 的网络策略。您可能需要添加网络策略以允许访问 EC2 IMDS。有关更多信息，请参阅 Amazon EC2 用户指南中的实例元数据和用户数据。

  对服务账户使用 IAM 角色的容器组（pod）无法访问 EC2 IMDS。
- Amazon VPC CNI plugin for Kubernetes 不会将网络策略应用于每个容器组的其他网络接口，而只应用于每个容器组的主接口（`eth0`）。这会影响以下架构：
  - `ENABLE_V4_EGRESS` 变量设置为 `true` 的 `IPv6` 容器组。此变量使 `IPv4` 出口功能能够将 IPv6 容器组连接到 `IPv4` 端点，例如集群外部的端点。`IPv4` 出口功能的工作原理是使用本地环回 IPv4 地址创建一个额外的网络接口。
  - 使用诸如 Multus 之类的链式网络插件时。由于这些插件为每个容器组添加了网络接口，因此网络策略不适用于链式网络插件。
- 默认情况下，网络策略功能使用节点上的端口 8162 获取指标。此外，该功能还使用端口 8163 进行运行状况探测。如果您在需要使用这些端口的节点或 Pod 内运行其他应用程序，则该应用程序将无法运行。在 VPC CNI 版本 `v1.14.1` 或更高版本中，您可以在以下位置更改这些端口：

AWS Management Console

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，然后选择要为其配置 Amazon VPC CNI 附加组件的集群名称。
3. 选择附加组件选项卡。
4. 选择附加组件框右上角的框，然后选择 Edit（编辑）。
5. 在 Configure ***name of addon*** （配置 name of addon）页面上：
   a. 在版本下拉列表中选择 `v1.14.0-eksbuild.3` 或更高版本。
   b. 展开可选配置设置。
   c. 在配置值中输入 JSON 键 `"enableNetworkPolicy":` 和值 `"true"`。生成的文本必须是有效的 JSON 对象。如果此键和值是文本框中的唯一数据，请用大括号 `{}` 将键和值括起来。

      以下示例启用了网络策略功能，启用了网络策略日志，将网络策略日志发送到 Amazon CloudWatch Logs，并且指标和运行状况探测设置为默认端口号：

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

如果您已通过 `helm` 安装 Amazon VPC CNI plugin for Kubernetes，则可以更新配置以更改端口。

- 运行以下命令以更改端口。分别在键 `nodeAgent.metricsBindAddr` 或键 `nodeAgent.healthProbeBindAddr` 的值中设置端口号。

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. 在编辑器中打开 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

2. 在 VPC CNI aws-node 守护程序集清单中 aws-network-policy-agent 容器的 args: 中，替换以下命令参数中的端口号。

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

先决条件

- 最低集群版本

现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。集群必须是 1.25 版本或更高版本的 Kubernetes。该集群必须运行下表中列出的 Kubernetes 版本和平台版本之一。请注意，所有比所列版本更高的 Kubernetes 和平台版本也受支持。您可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令来检查当前的 Kubernetes 版本：

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Kubernetes 版本	平台版本
1.27.4	eks.5
1.26.7	eks.6

Kubernetes 版本	平台版本
1.25.12	eks.7

- 最低 Amazon CNI 版本

  集群上 1.14 版本或更高版本的 Amazon VPC CNI plugin for Kubernetes。您可以使用以下命令查看当前使用的版本。

  ```
  kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
   cut -d : -f 3
  ```

  如果您的版本低于 1.14，请查看 更新 Amazon EKS 附加组件 将版本升级到 1.14 版本或更高版本。

- 最低 Amazon 内核版本

  您的节点必须具有 Linux 内核版本 5.10 或更高版本。您可以使用 uname -r 检查您的内核版本。如果您使用的是最新版本的 Amazon EKS 优化 Amazon Linux、Amazon EKS 优化加速的 Amazon Linux AMI 和 Bottlerocket AMI，则已拥有所需的内核版本。

  Amazon EKS 优化加速的 Amazon Linux AMI 版本 v20231116，或更高版本拥有内核版本 5.10。

将集群配置为使用 Kubernetes 网络策略

1. 挂载 BPF 文件系统

   > **Note**
   >
   > 如果您的集群是版本 1.27 或更高版本，则可以跳过此步骤，因为所有 Amazon EKS 优化版 Amazon Linux 和 Bottlerocket AMI 1.27 或更高版本都已具有此功能。
   > 对于所有其他集群版本，如果您将 Amazon EKS 优化版 Amazon Linux 升级到版本 v20230703 或更高版本，或者将 Bottlerocket AMI 升级到版本 v1.0.2 或更高版本，则可以跳过此步骤。

   a. 在每个节点上挂载 Berkeley Packet Filter（BPF）文件系统。

      ```
      sudo mount -t bpf bpffs /sys/fs/bpf
      ```

b. 然后，将相同的命令添加到 Amazon EC2 Auto Scaling 的启动模板中的用户数据。

2. 在 VPC CNI 中启用网络策略

a. 查看集群上当前安装的附加组件类型。根据您创建集群时使用的工具，您的集群上目前可能没有安装 Amazon EKS 附加组件类型。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

如果返回来的是版本号，则表明您的集群上安装有 Amazon EKS 类型的附加组件，而且此流程中其余的步骤，您也不需要走完。如果返回来的是一个错误，则表明您的集群上没有安装 Amazon EKS 类型的附加组件。

b. • Amazon EKS 附加组件

AWS Management Console

a. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。

b. 在左侧导航窗格中，选择集群，然后选择要为其配置 Amazon VPC CNI 附加组件的集群名称。

c. 选择附加组件选项卡。

d. 选择附加组件框右上角的框，然后选择 Edit（编辑）。

e. 在 Configure ***name of addon***（配置 name of addon）页面上：

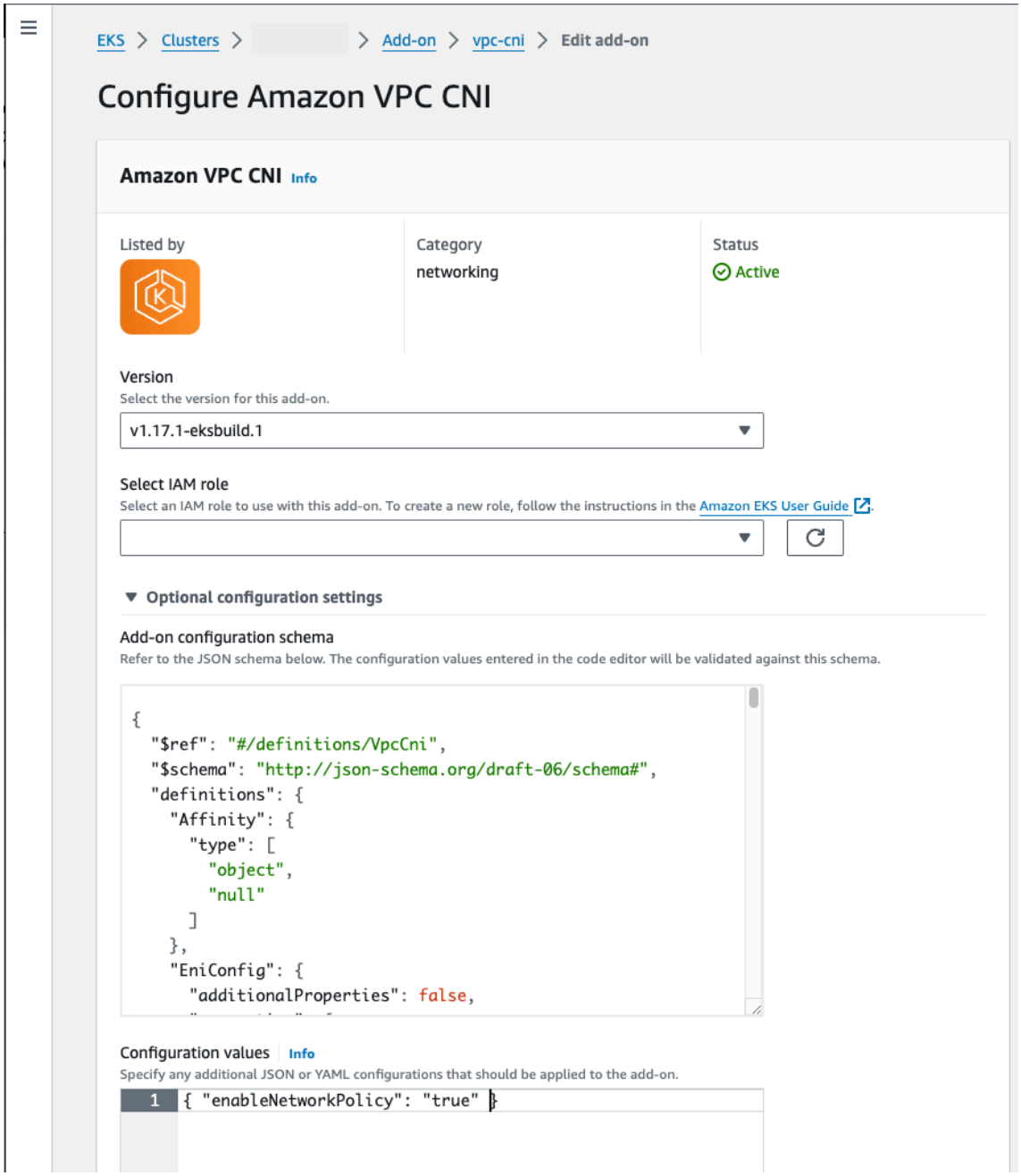
i. 在版本下拉列表中选择 `v1.14.0-eksbuild.3` 或更高版本。

ii. 展开可选配置设置。

iii. 在配置值中输入 JSON 键 `"enableNetworkPolicy":` 和值 `"true"`。生成的文本必须是有效的 JSON 对象。如果此键和值是文本框中的唯一数据，请用大括号 `{}` 将键和值括起来。以下示例显示网络策略已启用：

```
{ "enableNetworkPolicy": "true" }
```

下面的屏幕截图为此场景的一个示例。



AWS CLI

- 运行以下 AWS CLI 命令：将 `my-cluster` 替换为集群的名称，并将 IAM 角色 ARN 替换为您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni
 --addon-version v1.14.0-eksbuild.3 \
```

```
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values
 '{"enableNetworkPolicy": "true"}'
```

- 自行管理的附加组件

  Helm

  如果您已通过 helm 安装 Amazon VPC CNI plugin for Kubernetes，则可以更新配置以启用网络策略。

  - 运行以下命令以启用网络策略。

    ```
    helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace
     kube-system eks/aws-vpc-cni
    ```

  kubectl

  a. 在编辑器中打开 amazon-vpc-cni ConfigMap。

     ```
     kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
     ```

  b. 将以下行添加到 ConfigMap 中的 data。

     ```
     enable-network-policy-controller: "true"
     ```

     添加该行后，您的 ConfigMap 应该看起来像下面的示例。

     ```
     apiVersion: v1
      kind: ConfigMap
      metadata:
       name: amazon-vpc-cni
       namespace: kube-system
      data:
       enable-network-policy-controller: "true"
     ```

  c. 在编辑器中打开 aws-node DaemonSet。

```
kubectl edit daemonset -n kube-system aws-node
```

d. 在 VPC CNI aws-node 守护程序集清单中 aws-network-policy-agent 容器的 args: 中，将命令参数 --enable-network-policy=false 中的 false 替换为 true。

```
    - args:
       - --enable-network-policy=true
```

3. 确认 aws-node 容器组（pod）正在您的集群上运行。

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

示例输出如下。

```
aws-node-gmqp7                                          2/2     Running   1 (24h
 ago)   24h
aws-node-prnsh                                          2/2     Running   1 (24h
 ago)   24h
```

如果启用网络策略，则 aws-node 容器组（pod）中有 2 个容器。在以前的版本中，如果禁用网络策略，则 aws-node 容器组（pod）中只有 1 个容器。

您现在可以将 Kubernetes 网络策略部署到集群。有关更多信息，请参阅 Kubernetes 网络策略。

Stars 网络策略演示

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://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
```

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
```

2. 请查看集群上的所有 Pods。

```
kubectl get pods -A
```

示例输出如下。

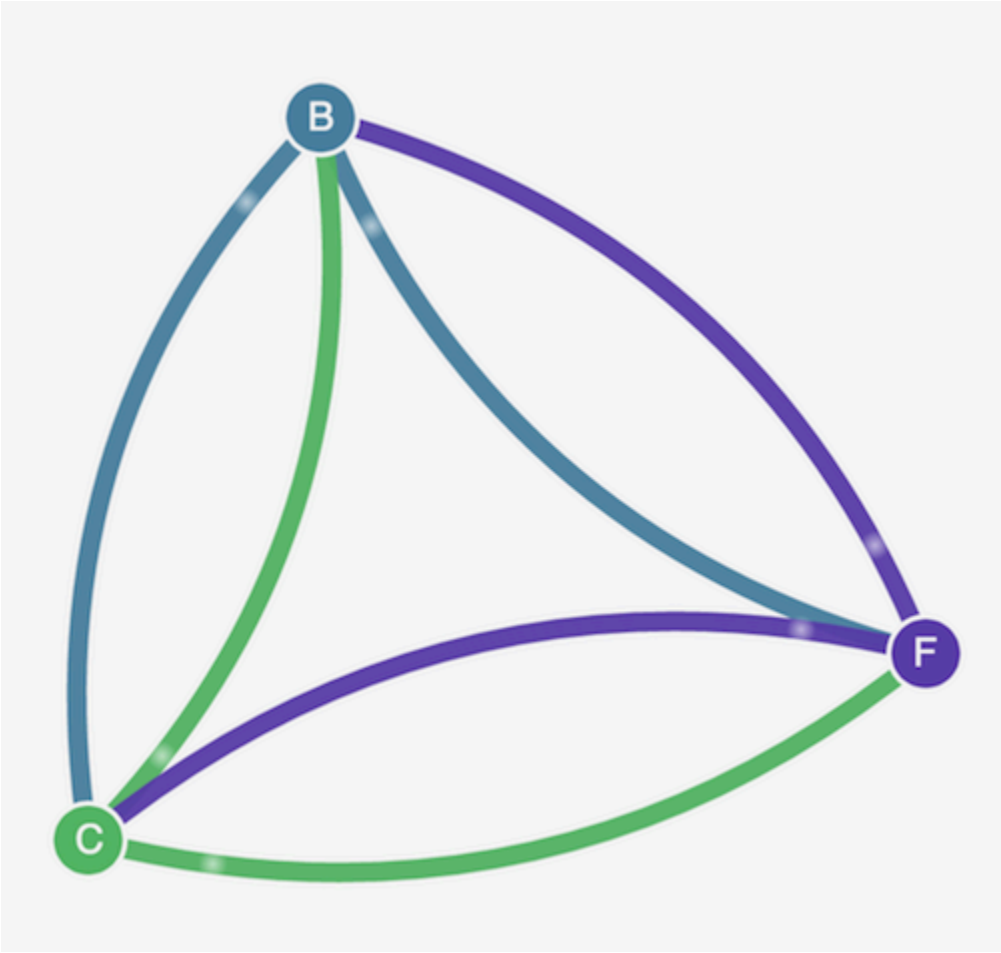
在输出中，您应该会在以下输出中显示的命名空间中看到容器。容器组（pod）的 *NAMES* 和 READY 列中的容器组（pod）数量与以下输出中的不同。在看到具有相似名称的容器并且它们在 STATUS 列中都具有 Running 之前，请不要继续。

```
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，请连接到集群上运行的 EXTERNAL-IP 服务：

```
kubectl get service/management-ui -n management-ui
```

4. 打开浏览器，进入上一步中的位置。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. 在 stars 和 client 命名空间中应用以下网络策略，以将服务彼此隔离：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

您可以使用以下命令将策略应用于两个命名空间：

```
kubectl apply -n stars -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
kubectl apply -n client -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/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. 应用以下不同的网络策略以允许管理用户界面访问这些服务。应用此策略以允许 UI：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

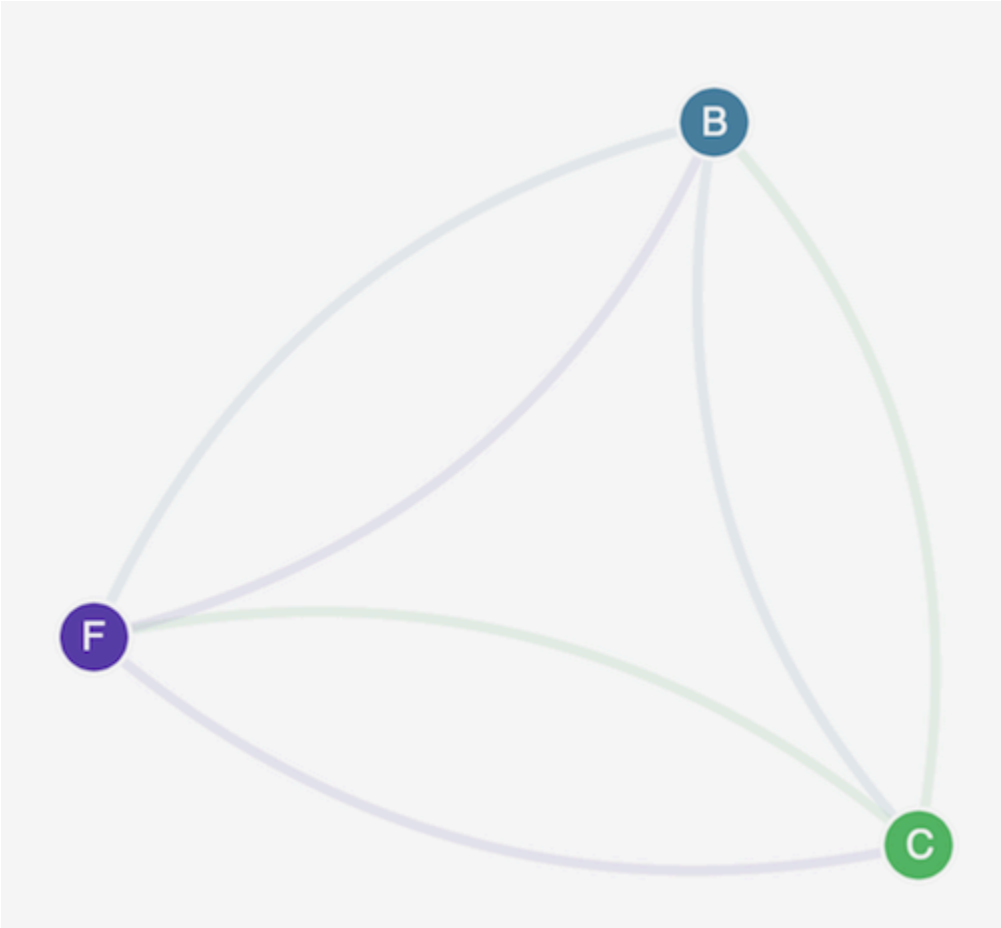
应用此策略以允许客户端：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

您可以使用以下命令应用这两个策略：

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/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。

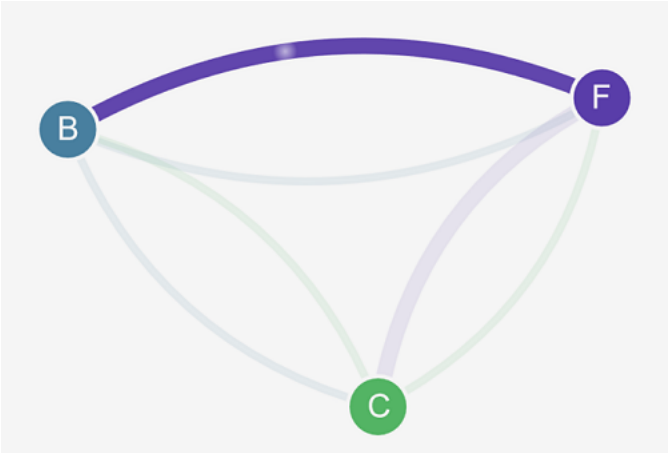


9. 应用以下网络策略以允许流量从前端服务传到后端服务：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
```

```
          port: 6379
```

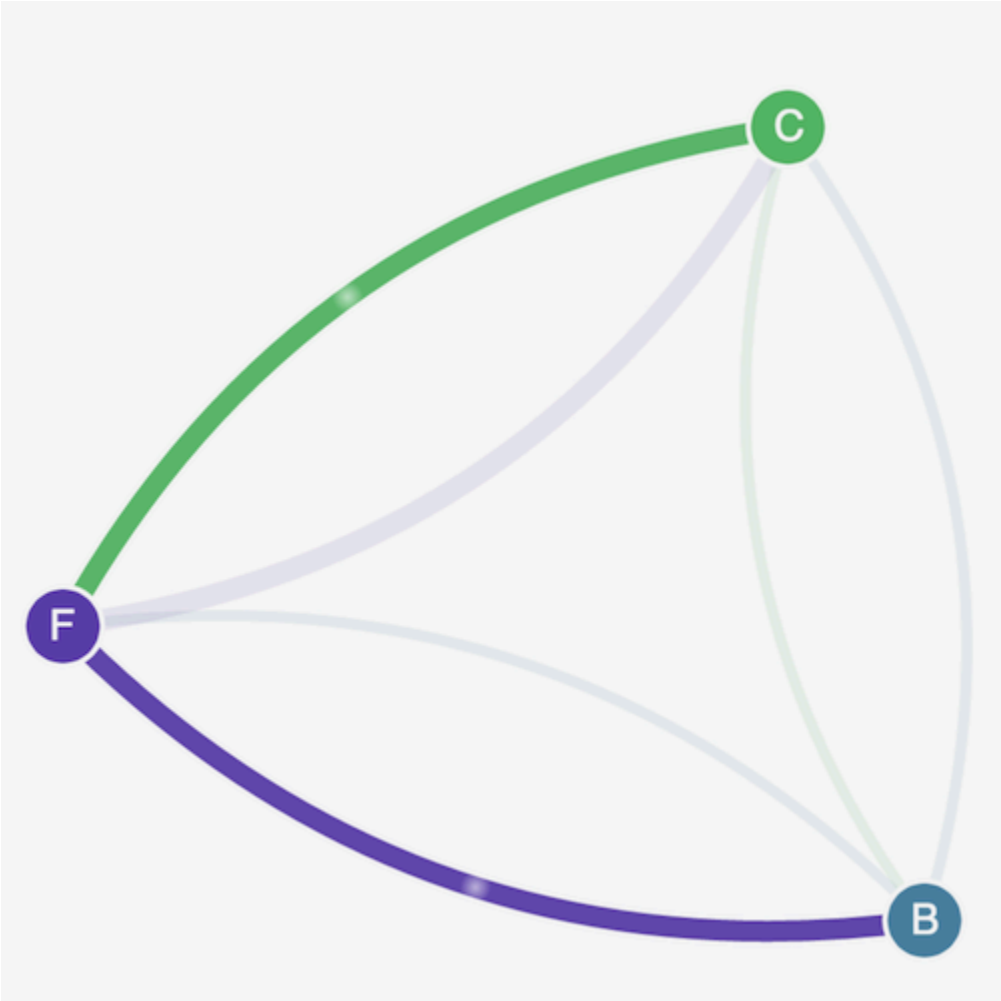
10. 刷新您的浏览器。您将看到前端可以与后端进行通信。



11. 应用以下网络策略以允许流量从客户端传到前端服务：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
          port: 80
```

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://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
```

甚至在删除资源后，节点上仍可能存在网络策略端点，这些端点可能会以意想不到的方式干扰集群中的网络。删除这些规则的唯一可靠方法是重新启动节点或终止所有节点并将它们回收。要终止所有节点，请将自动扩缩组所需计数设置为 0，然后创建所需数量的备份，或者仅终止节点。

排查网络策略问题

您可以通过查看 网络策略日志 和运行 eBPF 软件开发工具包 中的工具，对使用网络策略的网络连接进行问题排查和调查。

网络策略日志

网络策略是允许还是拒绝连接将记录在流日志中。每个节点上的网络策略日志包括每个具有网络策略的容器组（pod）流日志。网络策略日志存储在 /var/log/aws-routed-eni/network-policy-agent.log。以下示例来自 network-policy-agent.log 文件：

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

默认情况下，网络策略日志被禁用。要启用网络策略日志，请按照下列步骤操作：

> **Note**
>
> 网络策略日志需要为 VPC CNI aws-node 进程守护程序集清单中的 aws-network-policy-agent 容器额外提供 1 个 vCPU。

Amazon EKS 附加组件

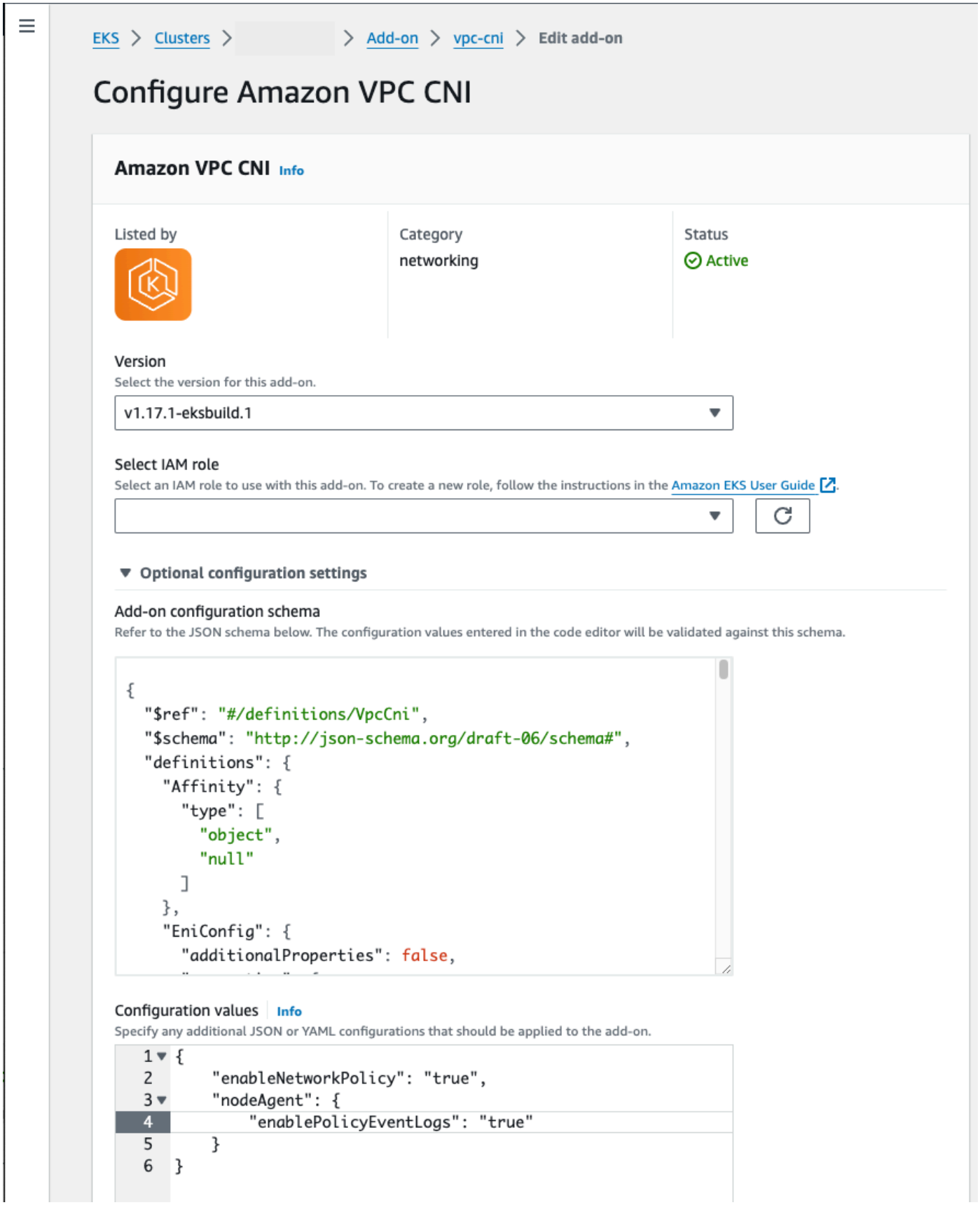
AWS Management Console

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，然后选择要为其配置 Amazon VPC CNI 附加组件的集群名称。
3. 选择附加组件选项卡。
4. 选择附加组件框右上角的框，然后选择 Edit（编辑）。
5. 在 Configure ***name of addon***（配置 name of addon）页面上：
   a. 在版本下拉列表中选择 v1.14.0-eksbuild.3 或更高版本。
   b. 展开可选配置设置。

c. 输入顶级 JSON 键 "nodeAgent":，值是一个在配置值中键为 "enablePolicyEventLogs": 且值为 "true" 的对象。生成的文本必须是有效的 JSON 对象。以下示例显示网络策略和网络策略日志已启用，并将网络策略日志发送到 CloudWatch Logs：

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

下面的屏幕截图为此场景的一个示例。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

- 运行以下 AWS CLI 命令：将 `my-cluster` 替换为集群的名称，并将 IAM 角色 ARN 替换为您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

自行管理的附加组件

Helm

如果您已通过 `helm` 安装 Amazon VPC CNI plugin for Kubernetes，则可以更新配置以编写网络策略日志。

- 运行以下命令以启用网络策略。

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

如果您已通过 `kubectl` 安装 Amazon VPC CNI plugin for Kubernetes，则可以更新配置以编写网络策略日志。

1. 在编辑器中打开 `aws-node DaemonSet`。

```
kubectl edit daemonset -n kube-system aws-node
```

2. 在 VPC CNI `aws-node` 守护程序集清单中 `aws-network-policy-agent` 容器的 `args:` 中，将命令参数 `--enable-policy-event-logs=false` 中的 `false` 替换为 `true`。

```
    - args:
       - --enable-policy-event-logs=true
```

将网络策略日志发送到 Amazon CloudWatch Logs

您可以使用 Amazon CloudWatch Logs 等服务监控网络策略日志。您可以使用以下方法将网络策略日志发送到 CloudWatch Logs。

对于 EKS 集群，策略日志将放在 /aws/eks/*cluster-name*/cluster/ 下；对于自行管理的 K8S 集群，日志将放在 /aws/k8s-cluster/cluster/ 下。

使用 Amazon VPC CNI plugin for Kubernetes 发送网络策略日志

如果启用网络策略，则会将第二个容器添加到节点代理的 aws-node 容器组（pod）。此节点代理可将网络策略日志发送到 CloudWatch Logs。

> **Note**
>
> 节点代理仅发送网络策略日志。不包括 VPC CNI 生成的其他日志。

先决条件

- 将以下权限作为节或单独的策略添加到您用于 VPC CNI 的 IAM 角色中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

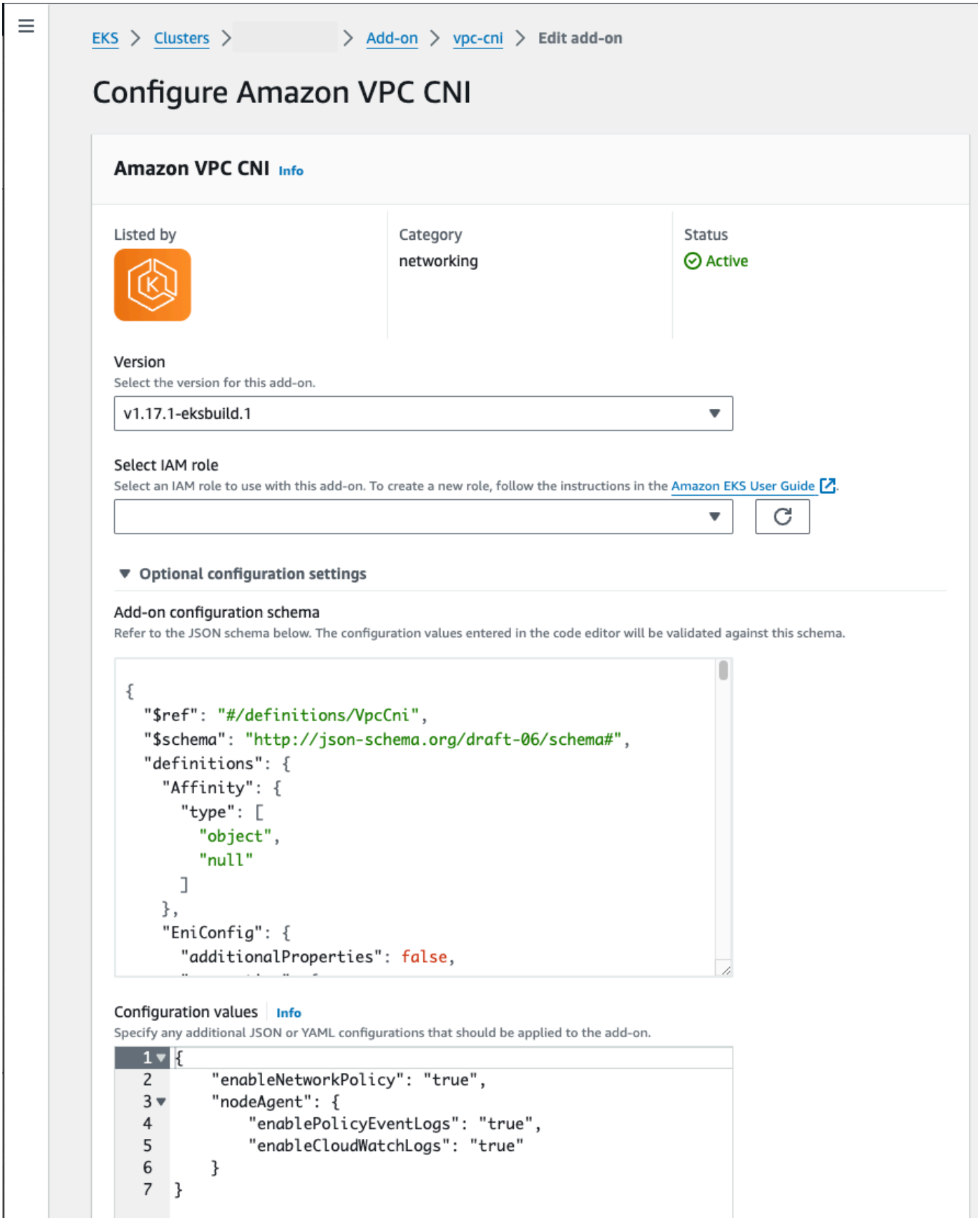
Amazon EKS 附加组件

AWS Management Console

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，然后选择要为其配置 Amazon VPC CNI 附加组件的集群名称。
3. 选择附加组件选项卡。
4. 选择附加组件框右上角的框，然后选择 Edit（编辑）。
5. 在 Configure ***name of addon***（配置 name of addon）页面上：
   a. 在版本下拉列表中选择 `v1.14.0-eksbuild.3` 或更高版本。
   b. 展开可选配置设置。
   c. 输入顶级 JSON 键 `"nodeAgent":`，值是一个在配置值中键为 `"enableCloudWatchLogs":` 且值为 `"true"` 的对象。生成的文本必须是有效的 JSON 对象。以下示例显示网络策略和网络策略日志已启用，并将日志发送到 CloudWatch Logs：

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

下面的屏幕截图为此场景的一个示例。


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

- 运行以下 AWS CLI 命令：将 `my-cluster` 替换为集群的名称，并将 IAM 角色 ARN 替换为您正在使用的角色。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

自行管理的附加组件

Helm

如果您已通过 `helm` 安装 Amazon VPC CNI plugin for Kubernetes，则可以更新配置以将网络策略日志发送到 CloudWatch Logs。

- 运行以下命令以启用网络策略日志并将它们发送到 CloudWatch Logs。

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. 在编辑器中打开 `aws-node DaemonSet`。

```
kubectl edit daemonset -n kube-system aws-node
```

2. 在 VPC CNI `aws-node` 进程守护程序集清单中 `aws-network-policy-agent` 容器的 `args:` 中，将两个命令参数 `--enable-policy-event-logs=false` 和 `--enable-cloudwatch-logs=false` 中的 `false` 替换为 `true`。

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

使用 Fluent Bit 守护程序集发送网络策略日志

如果您在守护进程集中使用 Fluent Bit 从节点发送日志，则可以添加配置以包含来自网络策略的网络策略日志。您可以使用以下示例配置：

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

已包含 eBPF SDK

Amazon VPC CNI plugin for Kubernetes 在节点上安装 eBPF SDK 工具集。您可以使用 eBPF SDK 工具来识别网络策略问题。例如，以下命令列出了节点上正在运行的程序。

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

要运行此命令，您可以使用任何方法连接到节点。

Kubernetes 网络策略

要实施 Kubernetes 网络策略，您需要创建 Kubernetes `NetworkPolicy` 对象并将它们部署到集群。`NetworkPolicy` 对象的范围限定到命名空间。您可以实施策略，根据标签选择器、命名空间和 IP 地址范围来允许或拒绝 Pods 之间的流量。有关创建 `NetworkPolicy` 对象的更多信息，请参阅 Kubernetes 文档中的网络策略。

Kubernetes `NetworkPolicy` 对象的执行是使用 Extended Berkeley Packet Filter (eBPF) 实施的。与基于 `iptables` 的实施相比，它具有更低的延迟和性能，包括降低 CPU 利用率和避免顺序查找。此外，eBPF 探测器还可以访问上下文丰富的数据，帮助调试复杂的内核级问题并提高可观测性。Amazon EKS 支持基于 eBPF 的导出器，该导出器利用探测器记录每个节点上的策略结果，并将数据导出到外部日志收集器以帮助调试。有关更多信息，请参阅 eBPF 文档。

容器组（pod）的自定义网络

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 集群、Pods 和 services 的 IPv6 地址。
- 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。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 您的设备或 AWS 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 1.29，则可以将 kubectl 的 1.28、1.29 或 1.30 版本与之配合使用。要安装或升级 kubectl，请参阅 安装或更新 kubectl。

- 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。此外，您的 Shell 的引用和转义规则可能有所不同。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的在 AWS CLI 中将引号和字符串结合使用。

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。

本主题中的命令使用使用 AWS CLI 示例中列出的惯例进行格式化。如果您从命令行运行命令，该命令行针对的资源位于与您正在使用的 AWS CLI 配置文件中定义的默认 AWS 区域不同的 AWS 区域，则需要向命令中添加 **--region *region-code***。

当您想将自定义联网部署到生产集群时，请跳至 步骤 2：配置 VPC。

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 创建 Amazon EKS 集群。

1. 定义一些要在其余步骤中使用的变量。

```
export cluster_name=my-custom-networking-cluster
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. 创建 VPC。

   1. 使用 Amazon EKS AWS CloudFormation 模板创建 VPC。

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/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
```

AWS 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 --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=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --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 $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --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 主体分配 `iam:CreateRole` 操作（权限）。

```
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 策略附加到角色。要将 IAM policy 附加到某个 IAM 主体，必须为附加该策略的主体分配以下 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 --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。

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 --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 --name my-custom-networking-cluster
```

步骤 2：配置 VPC

本教程需要在 步骤 1：创建测试 VPC 和集群 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. 确认您当前安装的 Amazon VPC CNI plugin for Kubernetes 为最新版本。要确定 Amazon EKS 附加组件类型的最新版本并将您的版本更新至此版本，请参阅 更新附加组件。要确定自行管理的附加组件类型的最新版本并将您的版本更新至此版本，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。
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 --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 \
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

   示例输出如下。

```
-----------------------------------
|          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 --cidr-block 192.168.1.0/24
```

3. 确认新区块已关联。

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

示例输出如下。

```
---------------------------------
|          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 --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 --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 --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

示例输出如下。

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  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。

> **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。
> - 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。

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
```

示例输出如下。

```
NAME         AGE
us-west-2a   117s
us-west-2d   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。

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（IRSA）。

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。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和自行管理节点组选项。

   - 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。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。

```
aws eks create-nodegroup --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 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 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 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'
```

> **Note**
>
> 如果您希望生产集群中的节点支持更高数量的 Pods，请在 Amazon EKS 建议每种 Amazon EC2 实例类型的最大 Pods 数量 中再次运行脚本。同时，将 **--cni-prefix-delegation-enabled** 选项添加到命令中。例如，将为 `m5.large` 实例类型返回 *110*。有关如何启用此功能的说明，请参阅 提高 Amazon EC2 节点的可用 IP 地址数量。您可以将此功能与自定义联网一起使用。

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
```

示例输出如下。

```
NAME                                          STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal   Ready    <none>   8m49s
 v1.22.9-eks-810597c
ip-192-168-0-92.us-west-2.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.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone:
 Placement.AvailabilityZone, SubnetId: SubnetId}'
```

示例输出如下。

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "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.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.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。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令：

      - 将 *my-cluster* 替换为您的集群的名称。
      - 将 *my-nodegroup* 替换为您的节点组的名称。

```
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
```

   示例输出如下。

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   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
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk            1/1     Running   0          7m15s
 192.168.0.126   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 --cluster-name $cluster_name --nodegroup-name my-nodegroup
```

即使在 AWS 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 --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 --name $cluster_name
```

通过以下命令确认集群已删除。

```
aws eks describe-cluster --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 --subnet-id $new_subnet_id_1
```

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

6. 删除您创建的 VPC。

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

提高 Amazon EC2 节点的可用 IP 地址数量

每个 Amazon EC2 实例支持最大数量的弹性网络接口以及可分配给每个网络接口的最大数量的 IP 地址。每个节点的每个网络接口都需要一个 IP 地址。可以将所有其他可用 IP 地址分配给 Pods。每个都 Pod 需要自己的 IP 地址。因此，您的节点可能有可用的计算和内存资源，但无法容纳其他 Pods，因为该节点已用完分配给 Pods 的 IP 地址。

在本主题中，您将学习如何通过分配 IP 前缀而不是为节点分配单个辅助 IP 地址来显著增加节点可以分配给 Pods 的 IP 地址数量。每个前缀都包含多个 IP 地址。如果您没有为集群配置 IP 前缀分配，则您的集群必须进行更多 Amazon EC2 应用程序编程接口（API）调用才能配置 Pod 连接所需的网络接口和 IP 地址。随着集群规模的扩大，这些 API 调用的频率可能会导致 Pod 和实例启动时间延长。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。有关更多信息，请参阅 GitHub 上的 Kubernetes 可扩展性阈值。

注意事项

- 每种 Amazon EC2 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。
- 默认情况下，可以在节点上运行的最大 Pods 数量为 110，但您可以更改该数字。如果您更改该数量并且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节点出现时使用更改后的值。
- 从分配 IP 地址过渡到分配 IP 前缀时，建议您创建新的节点组以增加可用 IP 地址的数量，而不是滚动替换现有节点。在同时分配 IP 地址和前缀的节点上运行 Pods 可能会导致通告的 IP 地址容量不一致，从而影响节点上未来的工作负载。有关执行过渡的推荐方式，请参阅 Amazon EKS 最佳实践指南中的在从辅助 IP 模式迁移到前缀委派模式期间替换所有节点（反之亦然）。
- 仅适用于具有 Linux 节点的集群。
  - 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 plugin for Kubernetes 附加组件版本降级到 `1.9.0`（或 `1.10.1`）以下。
  - 如果您的 Pods 在与 VPC 外部的端点通信的同时使用了 Pods 的安全组（`POD_SECURITY_GROUP_ENFORCING_MODE=standard`

和 AWS_VPC_K8S_CNI_EXTERNALSNAT=false），则使用节点的安全组，而不是您分配给 Pods 的任何安全组。

如果您的 Pods 在与 VPC 外部的端点通信的同时使用了 Pods 的安全组 (POD_SECURITY_GROUP_ENFORCING_MODE=strict)，则使用 Pod's 安全组。

先决条件

- 现有集群。要部署一个角色，请参阅 创建 Amazon EKS 集群。
- 您的 Amazon EKS 节点所在的子网必须有足够的连续 /28（适用于 IPv4 集群）或 /80（适用于 IPv6 集群）无类别域间路由 (CIDR) 块。IPv6 集群中只能有 Linux 节点。如果 IP 地址分散在整个子网 CIDR 中，则使用 IP 前缀可能会失败。建议执行下列操作：
  - 使用子网 CIDR 预留，这样即使保留范围内的任何 IP 地址仍在使用，在其释放后，这些 IP 地址也不会重新分配。这样可以确保前缀在不分段的情况下进行分配。
  - 使用专门用于运行分配 IP 前缀的工作负载的新子网。分配 IP 前缀时，Windows 和 Linux 工作负载可以在同一个子网中运行。
- 要为节点分配 IP 前缀，您的节点必须基于 AWS Nitro。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。
- 仅适用于具有 Linux 节点的集群 - 如果您的集群是针对 IPv4 系列配置的，则必须安装 Amazon VPC CNI plugin for Kubernetes 附加组件的 1.9.0 版或更高版本。您可以使用以下命令检查当前版本。

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

如果您的集群是针对 IPv6 系列配置的，则必须安装附加组件的 1.10.1 版。如果您的插件版本低于所需版本，则必须进行更新。想要了解更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件 的更新部分。

- 仅适用于具有 Windows 节点的集群
  - 您的集群及其平台版本必须等于或高于下表中的版本。要升级集群版本，请参阅 更新 Amazon EKS 集群 Kubernetes 版本。如果您的集群未达到最低平台版本，则在 Amazon EKS 更新您的平台版本之前，您无法为节点分配 IP 前缀。

Kubernetes 版本	平台版本
1.27	eks.3
1.26	eks.4
1.25	eks.5

您可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令来检查当前 Kubernetes 和平台版本：**aws eks describe-cluster --name *my-cluster* --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'**。

- 为集群启用 Windows 支持。有关更多信息，请参阅 为 Amazon EKS 集群启用 Windows 支持。

提高 Amazon EC2 节点的可用 IP 地址数量的步骤

1. 配置集群以将 IP 地址前缀分配给节点。在与节点的操作系统相匹配的选项卡上完成步骤。

   Linux

   1. 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

      即使您的子网有可用的 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。

      ```
      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。

2. 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 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。

3. 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
     ```

4. 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《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 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 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 容器组（pod）的自定义网络。

Windows

1. 启用 IP 前缀的分配。

   a. 打开 `amazon-vpc-cni` ConfigMap 进行编辑。

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. 将以下行添加到 `data` 部分。

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. 保存文件，然后关闭编辑器。

   d. 确认该行已添加到 ConfigMap。

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o
    "jsonpath={.data.enable-windows-prefix-delegation}"
   ```

   如果返回的输出并非 `true`，则可能存在错误。尝试再次完成该步骤。

   > **Important**
   >
   > 即使您的子网有可用的 IP 地址，如果子网没有任何连续 /28 数据块可用，您也将在节点事件中看到以下错误。
   >
   > ```
   > "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。

2. （可选）指定其他配置以控制集群的预扩展和动态扩展行为。有关更多信息，请参阅 GitHub 上的在 Windows 上使用前缀委派模式的配置选项。

   a. 打开 amazon-vpc-cni ConfigMap 进行编辑。

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. 用大于零的值替换 *example values*，然后将所需的条目添加到 ConfigMap 的 data 部分。如果您为 warm-ip-target 或 minimum-ip-target 设置了值，则该值将覆盖为 warm-prefix-target 设置的任何值。

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. 保存文件，然后关闭编辑器。

3. 创建至少有一种 Amazon EC2 Nitro 实例类型的 Windows 节点组。有关 Nitro 实例类型的列表，请参阅《Amazon EC2 用户指南》中的基于 Nitro 系统构建的实例。默认情况下，您可以部署到节点的最大 Pods 数量为 110。如果要增加或减少该数量，则请在引导配置的用户数据中指定以下内容。将 *max-pods-quantity* 替换为最大容器组值。

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

   如果您要部署托管节点组，则需要在启动模板中添加此配置。有关更多信息，请参阅使用启动模板自定义托管节点。有关 Windows 引导脚本配置参数的更多信息，请参阅引导脚本配置参数。

2. 节点部署完成后，请查看集群中的节点。

   ```
   kubectl get nodes
   ```

   示例输出如下。

   ```
   NAME                                                STATUS     ROLES    AGE   VERSION
   ip-192-168-22-103.region-code.compute.internal      Ready      <none>   19m   v1.XX.X-eks-6b7464
   ```

```
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-
eks-6b7464
```

3. 描述其中一个节点以确定该节点的 max-pods 值和可用 IP 地址数量。将 *192.168.30.193* 替换为之前输出中返回的其中一个节点名称中的 IPv4 地址。

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
PrivateIPv4Address'
```

示例输出如下。

```
pods:                                       110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

在先前的输出中，110 是 Kubernetes 将部署到节点的最大 Pods 数量，尽管有 *144* 个 IP 地址可用。

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：
- 适用于 Pods 的安全组不能与 Windows 节点一起使用。
- 通过使用版本 1.16.0 或更高版本的 Amazon VPC CNI 插件，适用于 Pods 的安全组可以与为包含 Amazon EC2 节点的 IPv6 系列配置的集群一起使用。通过使用版本 1.7.7 或更高版本的 Amazon VPC CNI 插件，您可以将适用于 Pods 的安全组与仅包含 Fargate 节点的集群配置 IPv6 系列一起使用。有关更多信息，请参阅 集群、Pods 和 services 的 IPv6 地址
- 大多数基于 Nitro 的 Amazon EC2 实例系列都支持 Pods 的安全组，但并非所有实例代的系列都支持安全组。例如，支持 m5、c5、r5、m6g、c6g 与 r6g 实例系列和实例代。但不支持 t 系列中的实例类型。有关支持的实例类型的完整列表，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 更改的访问权限，则必须在 Kubernetes ClusterRoleBinding 中为分配给您 psp 的 role 指定 eks:vpc-resource-controller

Kubernetes 用户。如果您使用的是默认 Amazon EKS psp、role 和 ClusterRoleBinding，则此命令为 eks:podsecuritypolicy:authenticated ClusterRoleBinding。例如，您将用户添加到 subjects: 部分，如以下示例所示：

```
[...]
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: eks:vpc-resource-controller
  - kind: ServiceAccount
    name: eks-vpc-resource-controller
```

- 如果您将自定义联网和适用于 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），则您向其分配安全组的 Pods 将不支持使用实例目标（externalTrafficPolicy 设置为 Local）的 NodePort 和 LoadBalancer 型 Kubernetes 服务。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
- 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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件

- 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 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件
- 对于具有高流失率的 Pods 而言，适用于 Pods 的安全组可能会导致更高的 Pod 启动延迟。这是由于资源控制器中的速率限制造成的。

为适用于 Pods 的安全组配置 Amazon VPC CNI plugin for Kubernetes

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。

1. 使用以下命令查看您当前的 Amazon VPC CNI plugin for Kubernetes 版本：

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
    | cut -d : -f 3
   ```

   示例输出如下。

   ```
   v1.7.6
   ```

   如果您的 Amazon VPC CNI plugin for Kubernetes 版本低于 `1.7.7`，请将该插件更新到版本 `1.7.7` 或更高版本。有关更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件

2. 将 `AmazonEKSVPCResourceController` 托管 IAM 策略添加到与您的 Amazon EKS 集群关联的集群角色。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. 检索您的集群 IAM 角色的名称，然后将其存储在一个变量中。将 *`my-cluster`* 替换为您的集群名称。

      ```
      cluster_role=$(aws eks describe-cluster --name my-cluster --query
       cluster.roleArn --output text | cut -d / -f 2)
      ```

   b. 将 策略附加到该角色。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

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，插件会为集群中的每个节点创建一个 cninode 自定义资源。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 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。

4. 您可以使用以下命令查看哪些节点具有 CNINode 自定义资源。如果返回 No resources found，则等几秒钟后重试。上一步需要重新启动 Amazon VPC CNI plugin for Kubernetes Pods，这需要几秒钟的时间。

```
$ kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal
 [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

如果您使用的是早于 1.15 的 VPC CNI 版本，则使用节点标签代替 CNINode 自定义资源。您可以使用以下命令查看哪些节点的节点标签 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 Amazon VPC CNI plugin for Kubernetes 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。

5. 如果您的集群使用 NodeLocal DNSCache，或者您想将 Calico 网络策略与拥有自己的安全组的 Pods 配合使用，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，并且必须启用以下设置：

```
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。在同一节点上使用不同安全组的容器组（pod）无法通信，因为它们配置在不同的子网中，且这些子网之间的路由被禁用。

- 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。

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 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. 创建要将资源部署到该处的 Kubernetes 命名空间。您可以将 *my-namespace* 替换为您要使用的命名空间名称。

```
kubectl create namespace my-namespace
```

2. 将 Amazon EKS SecurityGroupPolicy 部署到您的集群。

   a. 将以下内容复制到您的设备。如果您宁愿根据服务账户标签选择 Pods，则可以将 *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。您可以将 *my-role* 更改为您的角色名称。空的 serviceAccountSelector 会选择命名空间中的所有服务账户。您可以将 *my-security-group-policy* 替换为 SecurityGroupPolicy 的名称，并将 *my-namespace* 替换为要在其中创建 SecurityGroupPolicy 的命名空间。

   您必须将 *my_pod_security_group_id* 替换为现有安全组的 ID。如果您没有现有安全组，则必须创建一个安全组。有关更多信息，请参阅《Amazon EC2 用户指南》https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/中的适用于 Linux 实例的 Amazon EC2 安全组。您可以指定 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。

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
```

```
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

> **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`。
> - 这些安全组必须允许通过您为其配置探测器的任何端口，从应用于您节点的安全组（对于 kubelet）进行入站通信。
> - 这些安全组必须允许通过 TCP 和 UDP 端口 53 与分配到运行 CoreDNS 的 Pods（或 Pods 在其上运行的节点）的安全组进行出站通信。适用于您的 CoreDNS Pods 的安全组必须允许来自您指定的安全组的入站 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
```

3. 部署其标签与上一步中指定的 *podSelector* 的 *my-role* 值相匹配的示例应用程序。

a. 将以下内容复制到您的设备。将*###*替换为您自己的值，然后运行修改后的命令。如果您替换 *my-role*，请确保它与上一步中为选择器指定的值相同。

```
cat >sample-application.yaml <<EOF
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:1.23
        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
EOF
```

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
```

4. 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
```

示例输出如下。

```
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>
```

> **Note**
>
> - 如果任何 Pods 卡在 Waiting 状态，则运行 kubectl describe pod *my-deployment-xxxxxxxxxx-xxxxx* -n *my-namespace*。如果您看到了 Insufficient permissions: Unable to create Elastic Network Interface.，请确认您已在上一步中将 IAM 策略添加到 IAM 集群角色。
> - 如果任何 Pods 卡在 Pending 状态，请确认您的节点实例类型已在 limits.go 中列出，并且尚未达到实例类型支持的最大分支网络接口数与您的节点组中节点数的乘积。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 Amazon VPC CNI plugin for Kubernetes 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。该插件会记录此事件，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。有关如何删除网络接口的信息，请参阅《Amazon EC2 用户指南》中的删除网络接口。

5. 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
```

6. 从 TerminalB 的 shell 中，确认示例应用程序是否有效。

```
curl my-app
```

示例输出如下。

```
<!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。

7. 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。

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. 从 TerminalB 中，尝试再次访问该应用程序。

```
curl my-app
```

示例输出如下。

```
curl: (6) Could not resolve host: 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。

9. 完成实验后，您可以移除您创建的示例安全组策略、应用程序和安全组。从 TerminalA 运行以下命令。

```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
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。AWS 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 AWS 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。

备选的兼容 CNI 插件

Amazon VPC CNI plugin for Kubernetes 是 Amazon EKS 唯一支持的 CNI 插件。Amazon EKS 运行上游 Kubernetes，因此您可以将备选的兼容 CNI 插件安装到集群中的 Amazon EC2 节点。如果集群

中包含 Fargate 节点，则 Amazon VPC CNI plugin for Kubernetes 已在 Fargate 节点上。这是唯一可与 Fargate 节点结合使用的 CNI 插件。尝试在 Fargate 节点上安装备用 CNI 插件失败。

如果您计划在 Amazon EC2 节点上使用备用 CNI 插件，我们建议您获得插件的商业支持，或者利用内部专业知识对 CNI 插件项目进行故障排除并提供修复。

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。有关版本、资格和所执行测试的详细信息，请参阅以下合作伙伴的文档。

合作伙伴	产品	文档
Tigera	Calico	安装说明
Isovalent	Cilium	安装说明
Juniper	云原生 Contrail 联网（CN2）	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。

可选的兼容网络策略插件

Calico 是一种广泛采用的容器网络和安全解决方案。在 EKS 上使用 Calico 可为 EKS 集群提供完全合规的网络策略实施。此外，也可选择使用 Calico 的网络，便于保护底层 VPC 的 IP 地址。Calico Cloud 增强了 Calico Open Source 的功能，因此可提供更高的安全性与可观测性。

AWS Load Balancer Controller 是什么？

AWS Load Balancer Controller 管理适用于 Kubernetes 集群的 AWS 弹性负载均衡器。您可以使用控制器将您的集群应用程序公开到互联网。控制器预调配指向集群服务或入口资源的 AWS 负载均衡器。换句话说，控制器创建一个指向集群中多个容器组（pod）的 IP 地址或 DNS 名称。

控制器监视 Kubernetes Ingress 或 Service 资源。作为响应，它会创建相应的 AWS 弹性负载均衡资源。您可以通过对 Kubernetes 资源应用注释来配置负载均衡器的特定行为。例如，您可以使用注释将 AWS 安全组附加到负载均衡器。

此控制器预置以下资源：

Kubernetes `Ingress`

当您创建 Kubernetes `Ingress` 时，LBC 会创建 AWS 应用程序负载均衡器（ALB）。查看可以应用于入口资源的注释。

`LoadBalancer` 类型的 Kubernetes 服务

LBC 将在您创建 `LoadBalancer` 类型的 Kubernetes 服务时创建一个 AWS 网络负载均衡器（NLB）。查看可以应用于服务资源的注释。

在过去，实例目标使用 Kubernetes 网络负载均衡器，而 IP 目标使用 LBC。使用 AWS 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。

控制器是一个托管在 GitHub 上的开源项目。

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 和 Amazon EKS 上的网络负载均衡 的先决条件和注意事项。在这些主题中，您将部署包含 AWS 负载均衡器的示例应用程序。

安装控制器

- 了解如何 the section called “使用 Helm 安装”。如果您不熟悉 Amazon EKS，请使用此程序。此过程使用 Helm、Kubernetes 的程序包管理器和 `eksctl` 简化 LBC 的安装过程。
- 或者，the section called “使用清单安装”。此过程适用于高级集群配置。这包括对公共容器注册表具有有限网络访问权限的集群。

从已弃用的控制器版本迁移

- 如果您安装了 AWS Load Balancer Controller 的已弃用版本，则请学习如何 the section called “从已弃用的控制器迁移”。
- 已弃用的版本无法升级。必须将其移除并安装最新版本的 AWS Load Balancer Controller。
- 已弃用的版本包括：
  - Kubernetes 的 AWS ALB 入口控制器（“入口控制器”），是 AWS Load Balancer Controller 的前身。
  - AWS Load Balancer Controller 的任何 `0.1.`*x* 版本控制

传统云提供商

Kubernetes 包括的 AWS 传统云提供商。传统云提供商能够预调配 AWS 负载均衡器，这与 AWS Load Balancer Controller 类似。传统云提供商创建经典负载均衡器。如果您不安装 AWS Load Balancer Controller，则 Kubernetes 将默认设置为使用传统云提供商。您应该安装 AWS Load Balancer Controller 并避免使用传统云提供商。

> **Important**
>
> 在版本 2.5 及更高版本中，AWS Load Balancer Controller 成为具有 `type: LoadBalancer` 的 Kubernetes 服务资源的默认控制器，并为每个服务创建了一个 AWS 网络负载均衡器（NLB）。为此它将为服务创建一个变异的 Webhook，对于新的 `type: LoadBalancer` 服务，后者会将 `spec.loadBalancerClass` 字段设置为 `service.k8s.aws/nlb`。您可以关闭此功能，然后将 Helm 图表值 `enableServiceMutatorWebhook` 设置为 `false`，从而恢复将 传统云提供商 作为默认控制器。除非您关闭此功能，否则集群不会为您的服务预置新的经典负载均衡器。现有的经典负载均衡器将继续运行。

使用 Helm 安装 AWS Load Balancer Controller

本主题介绍如何使用 Helm（Kubernetes 的程序包管理器）和 `eksctl` 安装 AWS Load Balancer Controller。控制器安装了默认的选项。有关控制器的更多信息，包括使用注释对其进行配置的详细信息，请参阅 GitHub 上的 AWS Load Balancer Controller 文档。

在以下步骤中，将 *`example values`* 替换为您自己的值：。

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。

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。
- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- 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中列出的最低版本。
- 熟悉 AWS Elastic Load Balancing。有关更多信息，请参阅 Elastic Load Balancing 用户指南。
- 熟悉 Kubernetes 服务和传入资源。

- Helm 已在本地安装。

步骤 1：使用 **eksctl** 创建 IAM 角色

> **Note**
>
> 您只需为 AWS Load Balancer Controller 创建一个 IAM 角色，每个 AWS 账户一个。检查 IAM 控制台中是否存在 `AmazonEKSLoadBalancerControllerRole`。如果此角色存在，则请跳至 the section called “步骤 2：安装 AWS Load Balancer Controller”。

创建一个 IAM 策略。

1. 下载 AWS Load Balancer Controller 的 IAM 策略，该策略允许负载均衡器代表您调用 AWS API。

   AWS

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
   ```

   ```
   $ mv iam_policy_us-gov.json iam_policy.json
   ```

2. 使用上一步中下载的策略创建一个 IAM 策略。

   ```
   $ aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

   > **Note**
   >
   > 如果您在AWS Management Console中查看策略，则控制台会显示有关 ELB 服务的警告，但不会显示有关 ELB v2 服务的警告。之所以发生这种情况，是因为策略中的某些操作适用于 ELB v2，但不适用于 ELB。您可以忽略有关 ELB 的警告。

使用 **eksctl** 创建 IAM 角色

- 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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
$ 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
```

步骤 2：安装 AWS Load Balancer Controller

使用 Helm V3 安装 AWS Load Balancer Controller

1. 添加 `eks-charts` Helm 图表存储库。AWS 在 GitHub 上维护此存储库。

```
$ helm repo add eks https://aws.github.io/eks-charts
```

2. 更新您的本地存储库，以确保您拥有最新的图表。

```
$ helm repo update eks
```

3. 安装 AWS Load Balancer Controller。

   将 *my-cluster* 替换为您的集群名称。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。

   有关配置 Helm 图表的更多信息，请参阅 GitHub 上的 `values.yaml`。

```
$ 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
```

a. 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`***

b. 要查看 Helm 图表和负载均衡器控制器的可用版本，请使用以下命令：

```
helm search repo eks/aws-load-balancer-controller --versions
```

> **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，在上一个命令中将 *`install`* 更改为 **`upgrade`**。
> `helm install` 命令会自动安装控制器的自定义资源定义（CRDs）。`helm upgrade` 命令不会。如果您使用 `helm upgrade`，则必须手动安装 CRDs。运行以下命令来安装 CRDs：
>
> ```
> wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
> kubectl apply -f crds.yaml
> ```

步骤 3：验证控制器是否已安装

1. 验证控制器是否已安装。

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

示例输出如下。

```
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。

2. 在使用控制器预置AWS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 和Amazon EKS 上的网络负载均衡。

使用 Kubernetes 清单安装 AWS Load Balancer Controller 附加组件

本主题描述如何通过下载和应用 Kubernetes 清单安装控制器。您可以在 GitHub 上查看关于该控制器的完整文档。

在以下步骤中，将 *example values* 替换为您自己的值：。

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。

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。
- 集群的现有 AWS Identity and Access Management IAM OpenID Connect (OIDC) 提供商。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。
- 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中列出的最低版本。
- 熟悉 AWS Elastic Load Balancing。有关更多信息，请参阅 Elastic Load Balancing 用户指南。
- 熟悉 Kubernetes 服务和传入资源。

步骤 1：配置 IAM

> **Note**
>
> 您只需为 AWS Load Balancer Controller 创建一个 IAM 角色，每个 AWS 账户一个。检查 IAM 控制台中是否存在 `AmazonEKSLoadBalancerControllerRole`。如果此角色存在，则请跳至 the section called “步骤 2：安装 `cert-manager`”。

创建一个 IAM 策略。

1. 下载 AWS Load Balancer Controller 的 IAM 策略，该策略允许负载均衡器代表您调用 AWS API。

AWS

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
```

AWS GovCloud (US)

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
```

```
$ mv iam_policy_us-gov.json iam_policy.json
```

2. 使用上一步中下载的策略创建一个 IAM 策略。

```
$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> 如果您在AWS Management Console中查看策略，则控制台会显示有关 ELB 服务的警告，但不会显示有关 ELB v2 服务的警告。之所以发生这种情况，是因为策略中的某些操作适用于 ELB v2，但不适用于 ELB。您可以忽略有关 ELB 的警告。

eksctl

使用 **eksctl** 创建 IAM 角色

- 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。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。

```
$ 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
```

AWS CLI and kubectl

使用 AWS CLI 和 **kubectl** 创建 IAM 角色

1. 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。您需要为集群和 IAM 配置 OIDC。

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果返回输出，则您已有集群的 IAM OIDC 提供商。如果没有返回输出，则您必须为集群创建 IAM OIDC 提供商。有关更多信息，请参阅 为集群创建 IAM OIDC 提供商。

3. 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 AWS 区域。将 *EXAMPLED539D4633E53DE1B71EXAMPLE* 替换为上一步中返回的输出。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。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/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
```

```
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

4. 创建 IAM 角色。

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

5. 将所需的 Amazon EKS 托管 IAM 策略附加到 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
```

6. 将以下内容复制到您的设备。请将 *111122223333* 替换为您的账户 ID。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。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
```

7. 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
```

步骤 2：安装 **cert-manager**

使用以下方法之一安装 cert-manager 以将证书配置注入 Webhook。有关更多信息，请参阅 *cert-manager* 文档中的 入门。

我们建议使用 quay.io 容器注册表安装 cert-manager。如果您的节点无权访问 quay.io 容器注册表，则请使用 Amazon ECR 安装 cert-manager（见下文）。

Quay.io

使用 Quay.io 安装 **cert-manager**

- 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/v1.13.5/cert-manager.yaml
```

Amazon ECR

使用 Amazon ECR 安装 **cert-manager**

1. 使用以下方法之一安装 cert-manager 以将证书配置注入 Webhook。有关更多信息，请参阅 *cert-manager* 文档中的 入门。
2. 下载清单。

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

3. 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。

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

4. 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* 替换为您的私有注册表。

```
$ sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./
cert-manager.yaml
```

5. 运用该清单。

```
$ kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

步骤 3：安装 AWS Load Balancer Controller

使用 Kubernetes 清单安装 AWS Load Balancer Controller

1. 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_7_2_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.7.2/v2_7_2_full.yaml
```

2. 对文件进行以下编辑。

   a. 如果您已下载 v2_7_2_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 '596,604d' ./v2_7_2_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
---
```

b. 请将 *my-cluster* 替换为您的集群名称，将文件的 Deployment spec 部分中的 your-cluster-name 替换为您的集群名称。

```
$ sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_7_2_full.yaml
```

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。

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.7.2
```

请为清单添加注册表的名称。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，并将您的私有注册表名称添加到文件中。将 *111122223333.dkr.ecr.region-code.amazonaws.com* 替换为您的注册表。此行假定您的私有存储库名称与源存储库的名称相同。如果不相同，请将私有注册表名称后面的 eks/aws-load-balancer-controller 文本给更改为您的存储库名称。

```
$ sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-
controller|111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-
balancer-controller|' ./v2_7_2_full.yaml
```

d. （只有 Fargate 或受限的 IMDS 才需要此选项）

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
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. 应用文件。

```
$ kubectl apply -f v2_7_2_full.yaml
```

4. 将 IngressClass 和 IngressClassParams 清单下载到您的集群。

```
$ curl -Lo v2_7_2_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.7.2/v2_7_2_ingclass.yaml
```

5. 将清单应用于集群。

```
$ kubectl apply -f v2_7_2_ingclass.yaml
```

步骤 4：验证控制器是否已安装

1. 验证控制器是否已安装。

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

示例输出如下。

```
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。

2. 在使用控制器预置AWS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 和Amazon EKS 上的网络负载均衡。

从已弃用的控制器迁移

本主题介绍如何从已弃用的控制器版本迁移。更具体地说，它描述了如何移除 AWS Load Balancer Controller 的已弃用版本。

- 已弃用的版本无法升级。必须将其移除并安装最新版本的 LBC。
- 已弃用的版本包括：
  - Kubernetes 的 AWS ALB 入口控制器（“入口控制器”），是 AWS Load Balancer Controller 的前身。
  - AWS Load Balancer Controller 的任何 `0.1.x` 版本控制

移除已弃用的控制器版本

> **Note**
>
> 您可能已经使用 Helm 安装了已弃用的版本，或者使用 Kubernetes 清单手动安装了该版本。使用原来安装它的工具完成该过程。

使用 Helm 移除入口控制器

1. 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
```

2. 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
```

使用 Kubernetes 清单移除入口控制器

1. 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。

服务器错误（未找到）：未找到 deployments.apps “alb-ingress-controller”

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
```

2. 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
```

迁移到 AWS Load Balancer Controller

要从 Kubernetes 的 ALB 入口控制器迁移到 AWS Load Balancer Controller，您需要：

1. 移除 ALB 入口控制器（请参阅上述内容）。
2. 安装 AWS Load Balancer Controller。
3. 向 LBC 使用的 IAM 角色添加其他策略。此策略允许 LBC 管理由 Kubernetes 的 ALB 入口控制器创建的资源。

将迁移策略添加到 AWS Load Balancer Controller IAM 角色。

1. 下载该 IAM 策略。此策略允许 LBC 管理由 Kubernetes 的 ALB 入口控制器创建的资源。您还可以查看策略。

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.7.2/docs/install/iam_policy_v1_to_v2_additional.json
```

2. 如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 `arn:aws:` 替换为 `arn:aws-us-gov:`。。

```
$ sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

3. 创建 IAM 策略并记下返回的 ARN。

```
$ aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

4. 将 IAM 策略附加到 LBC 使用的 IAM 角色。将 *your-role-name* 替换为角色的名称，例如 AmazonEKSLoadBalancerControllerRole。

   如果您使用 eksctl 创建该角色，请找到已创建的角色名称，打开 AWS CloudFormation 控制台并选择 eksctl-*my-cluster*-addon-iamserviceaccount-kube-system-aws-load-balancer-controller 堆栈。选择资源选项卡。角色名称位于 Physical ID（物理 ID）列。如果您的集群位于 AWS GovCloud（美国东部）或 AWS GovCloud（美国西部）AWS 区域，则将 arn:aws: 替换为 arn:aws-us-gov:。

```
$ aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerAdditionalIAMPolicy
```

使用 CoreDNS Amazon EKS 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 AWS Fargate 配置文件 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。

下表列出了每个 Kubernetes 版本的 Amazon EKS 附加组件类型的最新版本。

Kubernetes 版本	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.11.1-e	v1.11.1-e	v1.10.1-e	v1.10.1-e	v1.9.3-ek	v1.9.3-ek	v1.9.3-ek	v1.8.7-ek

Kubernetes 版本	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	ksbuilc	ksbuilc	ksbuilc 1	ksbuilc 1	sbuild.	sbuild.	sbuild.	sbuild.10

> **Important**
>
> 如果您自行管理此附加组件，则表中的版本可能与可用的自行管理版本不同。有关更新此附加组件的自行管理类型的更多信息，请参阅更新自我管理的附加组件。

重要的 CoreDNS 升级注意事项

- 为了提高 CoreDNS Deployment 的稳定性和可用性，版本 v1.9.3-eksbuild.6 及更高版本和 v1.10.1-eksbuild.3 使用 PodDisruptionBudget 进行部署。如果您已部署现有 PodDisruptionBudget，则升级到这些版本可能会失败。如果升级失败，则完成以下任务之一应可解决问题：
  - 升级 Amazon EKS 附加组件时，选择覆盖现有设置作为冲突解决选项。如果您对 Deployment 进行了其他自定义设置，请务必在升级之前备份您的设置，以便在升级后可以重新应用其他自定义设置。
  - 删除现有的 PodDisruptionBudget，然后再次尝试升级。
- 在 EKS 附加组件版本 v1.9.3-eksbuild.3 及更高版本和 v1.10.1-eksbuild.6 及更高版本中，CoreDNS Deployment 将 readinessProbe 设置为使用 /ready 端点。此端点已在 CoreDNS 的 Corefile 配置文件中启用。

  如果您使用自定义 Corefile，则必须将 ready 插件添加到该配置，以便 /ready 端点在 CoreDNS 中处于活动状态，供探测器使用。
- 在 EKS 附加组件版本 v1.9.3-eksbuild.7 及更高版本 和 v1.10.1-eksbuild.4 及更高版本中，您可以更改 PodDisruptionBudget。您可以使用以下示例中的字段编辑附加组件并在可选配置设置中更改这些设置。此示例显示默认 PodDisruptionBudget。

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
```

```
}
```

您可以设置 maxUnavailable 或 minAvailable，但不能在一个 PodDisruptionBudget 中同时设置两者。有关 PodDisruptionBudgets 的更多信息，请参阅 Kubernetes 文档中的指定 PodDisruptionBudget。

请注意，如果将 enabled 设置为 false，则不会删除 PodDisruptionBudget。将此字段设置为 false 后，必须删除 PodDisruptionBudget 对象。同样，如果您在升级到带有 PodDisruptionBudget 的版本后编辑附加组件，以使用该附加组件的旧版本（降级附加组件），则不会删除 PodDisruptionBudget。要删除 PodDisruptionBudget，您可以运行以下命令：

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- 在 EKS 插件版本 v1.10.1-eksbuild.5 及更高版本中，将默认容差从 node-role.kubernetes.io/master:NoSchedule 更改为 node-role.kubernetes.io/control-plane:NoSchedule 以符合 KEP 2067。有关 KEP 2067 的更多信息，请参阅 GitHub 上的《Kubernetes 增强建议（KEP）》中的 KEP-2067：重命名 kubeadm“master”标签和污点。

  在 EKS 插件版本 v1.8.7-eksbuild.8 及更高版本和 v1.9.3-eksbuild.9 及更高版本中，两个公差都设置为与每个 Kubernetes 版本兼容。

- 在 EKS 附加组件版本 v1.9.3-eksbuild.11 和 v1.10.1-eksbuild.7 以及更高版本中，CoreDNSDeployment 为 topologySpreadConstraints 设置一个默认值。如果多个可用性区域内都有节点，则该默认值可确保 CoreDNS Pods 分布在可用性区域内。您可以设置一个用于代替默认值的自定义值。默认值如下所示：

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

CoreDNS v1.11 升级注意事项

- 在 EKS 附加组件版本 v1.11.1-eksbuild.4 以及更高版本中，容器映像基于 Amazon EKS Distro 维护的最小基本映像，其中包含最少的软件包且没有外壳。有关更多信息，请参阅 Amazon EKS Distro。CoreDNS 映像的使用和故障排查保持不变。

创建 Amazon EKS 附加组件

创建 Amazon EKS 类型的附加组件。Check

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。

1. 查看集群上当前安装的附加组件版本。

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
d : -f 3
```

示例输出如下。

```
v1.10.1-eksbuild.11
```

2. 查看集群上当前安装的附加组件类型。根据您创建集群时使用的工具，您的集群上目前可能没有安装 Amazon EKS 附加组件类型。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

如果返回来的是版本号，则表明您的集群上安装有 Amazon EKS 类型的附加组件，而且此流程中其余的步骤，您也不需要走完。如果返回来的是一个错误，则表明您的集群上没有安装 Amazon EKS 类型的附加组件。要安装，就需完成此流程中其余的步骤。

3. 保存您当前安装的附加组件的配置。

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. 使用 AWS CLI 创建附加组件。如果要使用 AWS Management Console 或 `eksctl` 来创建附加组件，请参阅 创建附加组件 并指定 `coredns` 为附加组件名称。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令。

   - 将 *my-cluster* 替换为您的集群名称。
   - 将 *v1.11.1-eksbuild.9* 替换为集群版本的最新版本（在最新版本表中列出）。

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-
version v1.11.1-eksbuild.9
```

如果您对当前附加组件应用的自定义设置与 Amazon EKS 附加组件的默认设置相冲突，则创建可能会失败。如果创建失败，您会收到一条可以帮助您解决问题的错误信息。或者，您可以将 **--resolve-conflicts OVERWRITE** 添加到前面的命令中。这样一来，附加组件会覆盖任何现有的自定义设置。创建附加组件后，您可以使用自定义设置对其进行更新。

5. 确认您的集群的 Kubernetes 版本的附加组件最新版本已添加到您的集群。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
```

附加组件创建可能需要几秒钟才能完成。

示例输出如下。

```
v1.11.1-eksbuild.9
```

6. 如果您在创建 Amazon EKS 附加组件之前对原始附加组件进行了自定义设置，则请使用您在上一步中保存的配置，以使用您的自定义设置更新 Amazon EKS 附加组件。

更新 Amazon EKS 附加组件

更新 Amazon EKS 类型的附加组件。如果您尚未将 Amazon EKS 类型的附加组件添加到集群中，则请添加它或查看 更新自我管理的附加组件，而不是完成此过程。

1. 查看集群上当前安装的附加组件版本。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
```

示例输出如下。

```
v1.10.1-eksbuild.11
```

如果返回的版本与最新版本表中集群的 Kubernetes 版本的版本相同，则您的集群上已经安装了最新版本，且您无需完成此过程的其余部分。如果您在输出中收到错误信息而不是版本号，则您的集群上没有安装 Amazon EKS 类型的附加组件。您需要先创建附加组件，然后才能使用此过程对其进行更新。

2. 保存您当前安装的附加组件的配置。

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. 使用 AWS CLI 更新您的附加组件。如果您想要使用 AWS Management Console 或 `eksctl` 更新附加组件，则请参阅 更新附加组件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令。

   - 将 *my-cluster* 替换为您的集群名称。
   - 将 *v1.11.1-eksbuild.9* 替换为集群版本的最新版本（在最新版本表中列出）。
   - **--resolve-conflicts** *PRESERVE* 选项保留附加组件的现有配置值。如果您为附加组件设置设定了自定义值，但未使用此选项，则 Amazon EKS 会使用其默认值覆盖您的值。如果您使用此选项，那么我们建议您在更新生产集群上的附加组件之前，先测试非生产集群上所有更改的字段和值。如果您将该值改为 `OVERWRITE`，则所有设置都将更改为 Amazon EKS 的默认值。如果您为任何设置设定了自定义值，这些值可能会被 Amazon EKS 的默认值覆盖。如果您将该值改为 `none`，Amazon EKS 不会更改任何设置的值，但更新可能会失败。如果更新失败，您会收到一条帮助您解决冲突的错误消息。
   - 如果您没有更新配置设置，则请从命令中移除 **--configuration-values '{*"replicaCount":3*}'**。如果您要更新配置设置，请将 *"replicaCount":3* 替换为要设置的设置。在此示例中，CoreDNS 的副本数设置为 3。您指定的值必须对配置架构有效。如果您不清楚配置架构，请运行 **aws eks describe-addon-configuration --addon-name coredns --addon-version *v1.11.1-eksbuild.9***，将 *v1.11.1-eksbuild.9* 替换为您要查看其配置的附加组件的版本号。将在输出中返回架构。如果您有任何现有自定义配置，想要将其全部删除，并将所有设置的值设置回 Amazon EKS 默认值，请从命令中删除 *"replicaCount":3*，这样就可以有空的 **{}**。有关 CoreDNS 设置的更多信息，请参阅 Kubernetes 文档中的自定义 DNS 服务。

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-
version v1.11.1-eksbuild.9 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

可能需要几秒钟才能完成更新。

4. 确认附加组件版本已更新。将 *my-cluster* 替换为您的集群名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

可能需要几秒钟才能完成更新。

示例输出如下。

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.1-eksbuild.9",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

更新自我管理的附加组件

> **Important**
>
> 建议您向集群添加 Amazon EKS 类型的附加组件，而不是自行管理类型的附加组件。如果不熟悉这些类型之间的区别，请参阅 the section called “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 the section called “创建附加组件”。如果您无法使用 Amazon EKS 附加组件，我们鼓励您向容器路线图 GitHub 存储库提交有关您为什么无法使用的问题。

1. 确认已在集群上安装自行管理类型的附加组件。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

如果返回错误消息，则表明集群上安装有自行管理类型的附加组件。完成此过程的其余步骤。如果返回版本号，则表明集群上安装有 Amazon EKS 类型的附加组件。要更新附加组件的 Amazon EKS 类型，请使用 更新 Amazon EKS 附加组件 中的过程，而不是此过程。如果不熟悉这些附加组件类型之间的区别，请参阅 Amazon EKS 附加组件。

2. 查看集群上当前安装的容器映像版本。

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

示例输出如下。

```
v1.8.7-eksbuild.2
```

3. 如果您的当前 CoreDNS 版本是 `v1.5.0` 或更高版本，但早于 CoreDNS 版本表中列出的版本，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。

   1. 使用以下命令打开 configmap。

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

   2. 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
      ```

4. 如果您最初在 Kubernetes `1.17` 或更低版本上部署了集群，那么您可能需要从 CoreDNS 清单中删除已停用的行。

   > **Important**
   >
   > 在更新到 CoreDNS 版本 `1.7.0` 之前，您必须完成此步骤，但是即使您要更新到早期版本，我们也建议您完成此步骤。

   1. 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。

2. 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
```

5. 检索您当前的 CoreDNS 映像版本：

```
kubectl describe deployment coredns -n kube-system | grep Image
```

示例输出如下。

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. 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
[...]
```

7. 通过将 *602401143452* 和 *region-code* 替换为上一步中返回的输出值来更新 CoreDNS 附件组件。将 *v1.11.1-eksbuild.9* 替换为您的 Kubernetes 版本的最新版本表中列出的 CoreDNS 版本。

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.1-
eksbuild.9
```

示例输出如下。

```
deployment.apps/coredns image updated
```

8. 再次检查容器映像版本，确认它已更新到您在上一步中指定的版本。

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

示例输出如下。

```
v1.11.1-eksbuild.9
```

自动扩缩 CoreDNS

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，默认情况下都会进行 CoreDNS 映像两个副本的 Deployment。这些 CoreDNS 容器组会为集群中的所有容器组提供名称解析。应用程序会通过名称解析连接到集群中的容器组和服务，并连接到集群外部的服务。随着容器组的名称解析（查询）请求数量增加，CoreDNS 容器组可能会不堪重负，继而减速，并拒绝容器组无法处理的请求。

要处理 CoreDNS 容器组上增加的负载，可以考虑使用 CoreDNS 的自动扩缩系统。Amazon EKS 可以在 CoreDNS 的 EKS 附加组件版本中管理 CoreDNS 部署的自动扩缩。此 CoreDNS 自动扩缩器会持续监控集群状态，包括节点数量和 CPU 核心数量。控制器会根据这些信息，动态调整 EKS 集群中 CoreDNS 部署的副本数量。此功能适用于 CoreDNS v1.9 和 EKS 发行版 1.25 及更高版本。有关与 CoreDNS 自动扩缩兼容的版本的更多信息，请参阅以下章节。

我们建议将此功能与其他 EKS 集群自动扩缩最佳实践结合使用，以提高应用程序的整体可用性和集群可扩展性。

先决条件

要让 Amazon EKS 扩展 CoreDNS 部署，需要满足三个先决条件：

- 您必须使用 CoreDNS 的 EKS 附加组件版本。
- 集群必须至少运行最低集群版本和平台版本。
- 集群必须至少运行最低 CoreDNS EKS 附加组件版本。

最低集群版本

由集群控制面板中的新组件完成 CoreDNS 自动扩缩，该组件由 Amazon EKS 管理。因此，您必须将集群升级到支持的最低平台版本（且其中包含新组件）的 EKS 版本。

新的 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。集群必须是 1.25 版本或更高版本的 Kubernetes。集群必须运行下表中列出的 Kubernetes 版本和平台版本之一或更高版本。请注意，所有比所列版本更高的 Kubernetes 和平台版本也受支持。您可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令来检查当前的 Kubernetes 版本：

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Kubernetes 版本	平台版本
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18
1.25.16	eks.19

> **Note**
>
> 还支持更高 Kubernetes 版本的每个平台版本，例如来自 eks.1 和之后的 Kubernetes 版本 1.30。

EKS 附加组件最低版本

Kubernetes 版本	1.29	1.28	1.27	1.26	1.25
	v1.11.1-eksbuild.9	v1.10.1-eksbuild.11	v1.10.1-eksbuild.11	v1.9.3-eksbuild.15	v1.9.3-eksbuild.15

在 AWS Management Console 中配置 CoreDNS 自动扩缩

1. 确保集群版本为最低版本或更高版本。

   Amazon EKS 会自动在同一 Kubernetes 版本的平台版本之间升级集群，您无法自行启动此过程。不过，您可以将集群升级到下一个 Kubernetes 版本，而后，集群将升级到该 K8s 版本和最新平台版本。例如，如果您从 `1.25` 升级为 `1.26`，集群将升级为 `1.26.15 eks.18`。

   新的 Kubernetes 版本有时会引入重大更改。因此，我们建议您先使用新的 Kubernetes 版本的单个集群来测试应用程序的行为，然后再更新生产集群。

   要将集群升级到新的 Kubernetes 版本，请按照 更新 Amazon EKS 集群 Kubernetes 版本 中的步骤操作。

2. 确保您具有 CoreDNS 的 EKS 附加组件，而不是自行管理的 CoreDNS 部署。

   根据您创建集群时使用的工具，您的集群上目前可能没有安装 Amazon EKS 附加组件类型。要查看集群上安装了哪种类型的附加组件，可以运行以下命令。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
   ```

   如果返回版本号，则表明集群上安装有 Amazon EKS 类型的附加组件，您可以继续执行下一步。如果返回来的是一个错误，则表明您的集群上没有安装 Amazon EKS 类型的附加组件。完成剩余的 创建 Amazon EKS 附加组件 步骤，将自行管理的版本替换为 Amazon EKS 附加组件。

3. 确保 CoreDNS 的 EKS 附加组件版本为 EKS 附加组件最低版本或更高版本。

查看集群上当前安装的附加组件版本。您可以在 AWS Management Console 中查看或运行以下命令：

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
```

示例输出如下。

```
v1.10.1-eksbuild.11
```

将此版本与上一节中的 EKS 附加组件最低版本进行比较。如果需要，请按照以下 更新 Amazon EKS 附加组件 步骤将 EKS 附加组件升级为更高版本。

4. 将自动扩缩配置添加到 EKS 附加组件的可选配置设置中。

   a. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。

   b. 在左侧导航窗格中，选择 Clusters（集群），然后选择要为其配置附加组件的集群的名称。

   c. 选择附加组件选项卡。

   d. 选择 CoreDNS 附加组件框右上角的框，然后选择编辑。

   e. 在配置 CoreDNS 页面上：

      i. 选择您想使用的 Version（版本）。我们建议您保留与上一步相同的版本，并通过单独的操作更新版本和配置。

      ii. 展开可选配置设置。

      iii. 在配置值中输入 JSON 键 "autoscaling":，以及键为 "enabled": 且值为 true 的嵌套 JSON 对象的值。生成的文本必须是有效的 JSON 对象。如果此键和值是文本框中的唯一数据，请用大括号 {} 将键和值括起来。以下示例显示了已启用的自动扩缩：

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

      iv. （可选）您可以提供自动扩缩所能扩展的 CoreDNS 容器组数量的最小值和最大值。

以下示例显示了已启用的自动扩缩以及所有带值的可选键。我们建议 CoreDNS 容器组的最小数量始终大于 2，以便为集群中的 DNS 服务提供弹性。

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. 要通过替换 CoreDNS 容器组来应用新配置，请选择保存更改。

Amazon EKS 通过推出适用于 CoreDNS 的 Kubernetes 部署，将更改应用到 EKS 附加组件。可在 AWS Management Console 中的附加组件更新历史记录中跟踪推出状态和 `kubectl rollout status deployment/coredns --namespace kube-system`。

`kubectl rollout` 具有以下命令：

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

如果推出时间过长，Amazon EKS 将撤销推出，并且会将类型为附加组件更新且状态为失败的消息添加到附加组件的更新历史记录中。要调查任何问题，请先查看推出历史记录，然后在 CoreDNS 容器组上运行 `kubectl logs`，进而查看 CoreDNS 的日志。

5. 如果更新历史记录中的新条目状态为成功，则表示推出已完成，并且附加组件正在所有 CoreDNS 容器组中使用新配置。更改集群中的节点数量和 CPU 核心数量时，Amazon EKS 会扩展 CoreDNS 部署的副本数量。

在 AWS Command Line Interface 中配置 CoreDNS 自动扩缩

1. 确保集群版本为最低版本或更高版本。

Amazon EKS 会自动在同一 Kubernetes 版本的平台版本之间升级集群，您无法自行启动此过程。不过，您可以将集群升级到下一个 Kubernetes 版本，而后，集群将升级到该 K8s 版本和最新平台版本。例如，如果您从 `1.25` 升级为 `1.26`，集群将升级为 `1.26.15 eks.18`。

新的 Kubernetes 版本有时会引入重大更改。因此，我们建议您先使用新的 Kubernetes 版本的单个集群来测试应用程序的行为，然后再更新生产集群。

要将集群升级到新的 Kubernetes 版本，请按照 更新 Amazon EKS 集群 Kubernetes 版本 中的步骤操作。

2. 确保您具有 CoreDNS 的 EKS 附加组件，而不是自行管理的 CoreDNS 部署。

   根据您创建集群时使用的工具，您的集群上目前可能没有安装 Amazon EKS 附加组件类型。要查看集群上安装了哪种类型的附加组件，可以运行以下命令。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
   ```

   如果返回版本号，则表明集群上安装有 Amazon EKS 类型的附加组件。如果返回来的是一个错误，则表明您的集群上没有安装 Amazon EKS 类型的附加组件。完成剩余的 创建 Amazon EKS 附加组件 步骤，将自行管理的版本替换为 Amazon EKS 附加组件。

3. 确保 CoreDNS 的 EKS 附加组件版本为 EKS 附加组件最低版本或更高版本。

   查看集群上当前安装的附加组件版本。您可以在 AWS Management Console 中查看或运行以下命令：

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   示例输出如下。

   ```
   v1.10.1-eksbuild.11
   ```

   将此版本与上一节中的 EKS 附加组件最低版本进行比较。如果需要，请按照以下 更新 Amazon EKS 附加组件 步骤将 EKS 附加组件升级为更高版本。

4. 将自动扩缩配置添加到 EKS 附加组件的可选配置设置中。

   运行以下 AWS CLI 命令：将 `my-cluster` 替换为集群的名称，并将 IAM 角色 ARN 替换为您正在使用的角色。

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   Amazon EKS 通过推出适用于 CoreDNS 的 Kubernetes 部署，将更改应用到 EKS 附加组件。可在 AWS Management Console 中的附加组件更新历史记录中跟踪推出状态和 `kubectl rollout status deployment/coredns --namespace kube-system`。

   `kubectl rollout` 具有以下命令：

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   undo     -- Undo a previous rollout
   ```

   如果推出时间过长，Amazon EKS 将撤销推出，并且会将类型为附加组件更新且状态为失败的消息添加到附加组件的更新历史记录中。要调查任何问题，请先查看推出历史记录，然后在 CoreDNS 容器组上运行 `kubectl logs`，进而查看 CoreDNS 的日志。

5. （可选）您可以提供自动扩缩所能扩展的 CoreDNS 容器组数量的最小值和最大值。

   以下示例显示了已启用的自动扩缩以及所有带值的可选键。我们建议 CoreDNS 容器组的最小数量始终大于 2，以便为集群中的 DNS 服务提供弹性。

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}, "minReplicas": 2, "maxReplicas": 10}'
   ```

6. 运行以下命令，检查附加组件的更新状态：

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns \
```

如果您看到此行："status": "ACTIVE"，则表示部署已完成，并且附加组件正在所有 CoreDNS 容器组中使用新配置。更改集群中的节点数量和 CPU 核心数量时，Amazon EKS 会扩展 CoreDNS 部署的副本数量。

CoreDNS 指标

CoreDNS 作为 EKS 插件，在 kube-dns 服务中以 Prometheus 格式公开端口 9153 上 CoreDNS 的指标。您可以使用 Prometheus、Amazon CloudWatch 代理，或任何其他兼容系统来抓取（收集）这些指标。

有关同时兼容 Prometheus 和 CloudWatch 代理的抓取配置示例，请参阅《Amazon CloudWatch 用户指南》中的适用于 Prometheus 的 CloudWatch 代理配置。

使用 Kubernetes **kube-proxy** 附加组件

> **Important**
>
> 建议您向集群添加 Amazon EKS 类型的附加组件，而不是自行管理类型的附加组件。如果不熟悉这些类型之间的区别，请参阅 the section called “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 the section called “创建附加组件”。如果您无法使用 Amazon EKS 附加组件，我们鼓励您向容器路线图 GitHub 存储库提交有关您为什么无法使用的问题。

kube-proxy 附加组件部署在 Amazon EKS 集群中的每个 Amazon EC2 节点上。该附加组件会在节点上维护网络规则，并实现与 Pods 的网络通信。该附加组件不会部署到集群中的 Fargate 节点。有关更多信息，请参阅 Kubernetes文档中的 kube-proxy。

下表列出了每个 Kubernetes 版本的 Amazon EKS 附加组件类型的最新版本。

Kubernetes 版本	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.30.0-eksbuild	v1.29.3-eksbuild	v1.28.8-eksbuild	v1.27.15-eksbuild.5	v1.26.15-eksbuild.5	v1.25.16-eksbuild.8	v1.24.17-eksbuild.8	v1.23.17-eksbuild.9

> **Important**
>
> 该文档的早期版本不正确。kube-proxy 版本 v1.28.5、v1.27.9 和 v1.26.12 不可用。如果您自行管理此附加组件，则表中的版本可能与可用的自行管理版本不同。

每个 Amazon EKS 集群版本都有两种类型的 kube-proxy 容器映像可用：

- 默认：此映像类型以 Kubernetes 上游社区维护的基于 Debian 的 Docker 映像为基础。
- 最低：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。

每个 Amazon EKS 集群版本的最新可用的自行管理 **kube-proxy** 容器映像版本

映像类型	1.30	1.29	1.28	1.27	1.26	1.25	1.24	1.23
kube-proxy （默认类型）	只有最小类型可用	只有最小类型可用	只有最小类型可用	只有最小类型可用	只有最小类型可用	只有最小类型可用	v1.24.17-eksbuild.2	v1.23.16-eksbuild.2
kube-proxy （最低要求类型）	v1.30.0-minimal-eksbuild.	v1.29.3-minimal-eksbuild.	v1.28.8-minimal-eksbuild.	v1.27.15-minimal-eksbuild	v1.26.15-minimal-eksbuild	v1.25.16-minimal-eksbuild	v1.24.17-minimal-eksbuild	v1.23.17-minimal-eksbuild.5

> **Important**
>
> - 默认镜像类型不适用于 Kubernetes 版本 1.25 及更高版本。您必须使用最小的镜像类型。
> - 更新 Amazon EKS 附加组件类型时，可以指定有效的 Amazon EKS 附加组件版本，该版本可能不是此表中列出的版本。这是因为 Amazon EKS 附加组件版本并不总是与更新此附加组件的自行管理类型时指定的容器映像版本相匹配。更新此附加组件的自行管理类型时，可指定此表中列出的有效容器映像版本。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。

注意事项

- Amazon EKS 集群上的 Kube-proxy 具有与 Kubernetes 相同的兼容性和偏斜策略。了解如何检索插件版本兼容性。
- Kube-proxy 必须与您的Amazon EC2 节点上的 kubelet 具有相同的次要版本。
- Kube-proxy 不能高于集群控制面板的次要版本。
- 如果您最近将集群更新到新的 Kubernetes 次要版本，请先将 Amazon EC2 节点更新到相同的次要版本，然后再将 kube-proxy 更新到与节点相同的次要版本。

更新 **kube-proxy** 自行管理的附加组件

1. 确认已在集群上安装自行管理类型的附加组件。将 *my-cluster* 替换为您集群的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 addon.addonVersion --output text
```

   如果返回错误消息，则表明集群上安装有自行管理类型的附加组件。本主题中的其余步骤用于更新自行管理类型的附加组件。如果返回版本号，则表明集群上安装有 Amazon EKS 类型的附加组件。要对其进行更新，请使用 更新附加组件 中的步骤，而不是本主题中的步骤。如果不熟悉这些附加组件类型之间的区别，请参阅 Amazon EKS 附加组件。

2. 查看集群上当前安装的容器映像版本。

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

示例输出如下。

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-eksbuild.2
```

在示例输出中，集群上安装的版本是 *v1.29.1-eksbuild.2*。

3. 将 *602401143452* 和 *region-code* 替换为上一步中输出的值来更新 kube-proxy 附加组件。将 *v1.30.0-eksbuild.3* 替换为每个 Amazon EKS 集群版本的最新可用的自行管理 kube-proxy 容器映像版本表中列出的 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=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.0-eksbuild.3
```

示例输出如下。

```
daemonset.apps/kube-proxy image updated
```

4. 确认新版本现已安装在集群上。

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

示例输出如下。

```
v1.30.0-eksbuild.3
```

5. 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，您无需在每次更新附加组件时都进行添加。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
```

6. 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 或更低版本创建的，并打算在您的集群中使用 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 添加到清单中之后，您无需在每次更新附加组件时都进行添加。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
   ```

使用接口端点访问 Amazon Elastic Kubernetes Service（AWS PrivateLink）

您可以使用 AWS PrivateLink 在您的 VPC 和 Amazon Elastic Kubernetes Service 之间创建私有连接。您可以像在 VPC 中一样访问 Amazon EKS，而无需使用互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。VPC 中的实例不需要公有 IP 地址即可访问 Amazon EKS。

您可以通过创建由 AWS PrivateLink 提供支持的接口端点来建立此私有连接。我们将在您为接口端点启用的每个子网中创建一个端点网络接口。这些是请求者托管式网络接口，用作发往 Amazon EKS 的流量的入口点。

有关更多信息，请参阅《AWS PrivateLink 指南》中的通过 AWS PrivateLink 访问 AWS 服务。

Amazon EKS 注意事项

- 在为 Amazon EKS 设置接口端点之前，请首先检查《AWS PrivateLink 指南》中的 注意事项。
- Amazon EKS 支持通过接口端点调用其所有 API 操作，但不支持调用 Kubernetes API。Kubernetes API 服务器已经支持私有端点。Kubernetes API 服务器私有端点为您用于与集群进行通信的 Kubernetes API 服务器创建私有端点（使用 Kubernetes 管理工具，如 `kubectl`）。您可以启用对 Kubernetes API 服务器的私有访问，以便您的节点与 API 服务器之间的所有通信都在 VPC 内。Amazon EKS API 的 AWS PrivateLink 可帮助您从 VPC 调用 Amazon EKS API，而无需向公共互联网公开流量。
- 您无法将 Amazon EKS 配置为只能通过接口端点进行访问。
- AWS PrivateLink 的标准定价适用于 Amazon EKS 的接口端点。计费方式：按在每个可用区中预置的接口端点每小时以及通过接口端点处理的数据。有关更多信息，请参阅AWS PrivateLink 定价。
- Amazon EKS 不支持 VPC 端点策略。默认情况下，允许通过接口端点对 Amazon EKS 进行完全访问。或者，您可以将安全组与端点网络接口关联，以控制通过接口端点流向 Amazon EKS 的流量。
- 您可以使用 VPC 流日志捕获有关传入和传出网络接口（包括接口端点）的 IP 流量的信息。您可以将流日志数据发布到 Amazon CloudWatch 或 Amazon S3。有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 流日志记录 IP 流量。
- 您可以从本地数据中心访问 Amazon EKS API，方法是将本地数据中心连接到具有接口端点的 VPC。您可以使用 AWS Direct Connect 或 AWS Site-to-Site VPN 将您的本地站点连接到 VPC。
- 您可以使用 AWS Transit Gateway 或 VPC 对等连接通过接口端点将其他 VPC 连接到 VPC。VPC 对等连接是两个 VPC 之间的网络连接。您可以在您的 VPC 之间建立 VPC 对等连接，或者在您的 VPC 与其他账户中的 VPC 之间建立此连接。VPC 可以位于不同的 AWS 区域。对等 VPC 之间的流量保留在 AWS 网络上。流量不会穿越公共互联网。中转网关是网络中转中心，您可用它来互连 VPC。VPC 和中转网关之间的流量仍保留在 AWS 全球私有网络上。流量不会在公共互联网上公开。
- Amazon EKS 的 VPC 接口端点只能通过 `IPv4` 访问。不支持 `IPv6`。
- 亚太地区（海得拉巴）、亚太地区（墨尔本）、亚太地区（大阪）、加拿大西部（卡尔加里）、欧洲（西班牙）、欧洲（苏黎世）或中东（阿联酋）AWS 区域 不支持 AWS PrivateLink。

为 Amazon EKS 创建接口端点

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface（AWS CLI）为 Amazon EKS 创建接口端点。有关更多信息，请参阅 AWS PrivateLink 指南中的创建 VPC 端点。

使用以下服务名称为 Amazon EKS 创建接口端点：

```
com.amazonaws.region-code.eks
```

在为 Amazon EKS 和其他 AWS 服务创建接口端点时，默认启用私有 DNS 功能。但是，您必须确保将以下 VPC 属性设置为 `true`：`enableDnsHostnames` 和 `enableDnsSupport`。有关更多信息，请参阅《Amazon VPC 用户指南》中的查看和更新 VPC 的 DNS 属性。为接口端点启用私有 DNS 功能后：

- 您可以使用 Amazon EKS 的默认区域 DNS 名称向其发出任何 API 请求。例如，`eks.region.amazonaws.com`。有关 API 列表，请参阅 Amazon EKS API Reference（《Amazon EKS API 参考》）中的操作。
- 您无需对调用 EKS API 的应用程序进行任何更改。
- 对 Amazon EKS 默认服务端点的任何调用都会通过接口端点自动路由到私有 AWS 网络上。

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 AWS Management Console 查看在您的每个集群上运行的工作负载的相关信息。
- 使用 Kubernetes Vertical Pod Autoscaler 纵向扩缩 Pods。
- 使用 Kubernetes Horizontal Pod Autoscaler 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 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。
- 您可以使用 `externalIPs` 限制可分配给服务的 IP 地址。

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 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。
- 安装在计算机上的 `Kubectl`。有关更多信息，请参阅 安装或更新 `kubectl`。
- `Kubectl` 配置为与集群通信。有关更多信息，请参阅 为 Amazon EKS 集群创建或更新 `kubeconfig` 文件。
- 如果计划将示例工作负载部署到 Fargate，现有 Fargate 配置文件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 AWS Fargate，请确保 AWS Fargate 配置文件 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。

      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:1.23
        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 AWS 服务交互，我们建议为 Pods 创建 Kubernetes 服务账户，然后将其关联到 AWS 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。

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 AWS Load Balancer Controller 以负载均衡服务的应用程序或网络流量。

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
```

示例输出如下。

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 和 Vertical Pod Autoscaler 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
```

示例输出如下。

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，后者是属于服务的一部分。

6. 在上一步查看命名空间时，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.us-west-2.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.us-west-2.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 容器组（pod）的自定义网络。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 AWS 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。

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
```

示例输出如下。

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

Windows

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

示例输出如下。

```
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
```

示例输出如下。

```
nameserver 10.100.0.10
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.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
```

后续步骤

部署示例应用程序后，您可能想尝试以下其中一些练习：

- the section called “应用程序负载均衡”
- the section called “网络负载均衡”

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。
- 您已安装 Kubernetes Metrics Server。有关更多信息，请参阅安装 Kubernetes Metrics Server。
- 您使用的是配置为与 Amazon EKS 集群通信的 `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 `registry.k8s.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。

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
```

```
registry.k8s.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 `registry.k8s.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 AWS 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/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./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
```

示例输出如下。

```
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
```

示例输出如下。

```
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
```

示例输出如下。

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.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
```

示例输出如下。

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.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
```

示例输出如下。

```
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。
- 您已安装 Kubernetes Metrics Server。有关更多信息，请参阅安装 Kubernetes Metrics Server。
- 您使用的是配置为与 Amazon EKS 集群通信的 `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
   ```

   示例输出如下。

   ```
   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
```

示例输出如下。

```
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
```

示例输出如下。

```
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 AWS 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。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 AWS 网站上的 Elastic Load Balancing 功能。

创建类型为 `LoadBalancer` 的 Kubernetes `Service` 时，AWS 云提供程序负载均衡器控制器默认创建 AWS 经典负载均衡器，但也可以创建 AWS 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 AWS 云提供程序负载均衡器的更多信息，请参阅 Kubernetes 文档中的 AWS 云提供程序负载均衡器控制器。本主题中未涵盖其使用。

我们建议您使用版本 `2.7.2` 或更高版本的 AWS Load Balancer Controller，而不是 AWS 云提供程序负载均衡器控制器。AWS Load Balancer Controller 创建 AWS 网络负载均衡器，但不创建 AWS 经典负载均衡器。本主题的其余部分介绍如何使用 AWS 负载均衡器控制器。

AWS 网络负载均衡器可以对部署到 Amazon EC2 IP 和实例目标或 AWS 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 AWS负载均衡器控制器。

先决条件

必须满足以下要求，才能使用 AWS 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。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。

- 在集群上部署 AWS Load Balancer Controller。有关更多信息，请参阅 AWS Load Balancer Controller 是什么？。建议升级到版本 2.7.2 或更高版本。
- 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 AWS 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 AWS 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 AWS 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 AWS 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 AWS 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 AWS AWS 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 AWS AWS CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC。
    - 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 AWS 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 AWS CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC。
    - 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。AWS 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 AWS Load Balancer Controller 和使用 AWS 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 AWS Load Balancer Controller 的注释。
- 使用 Amazon VPC CNI plugin for Kubernetes 时，AWS 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 AWS Load Balancer Controller 文档中的 AWS 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AWS Load Balancer Controller 文档。

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations:
 eipalloc-xxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- 对于您创建的每个网络负载均衡器，Amazon EKS 会向节点的安全组添加一个用于客户端流量的入站规则，并为 VPC 中的每个负载均衡器子网添加一个用于运行状况检查的规则。如果 Amazon EKS 尝试创建的规则超过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。AWS 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。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 AWS Load Balancer Controller 创建新的网络负载均衡器。尝试替换使用 AWS 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 AWS Load Balancer Controller 而不是 AWS 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。

```
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

AWS 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 AWS 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 AWS 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。`aws-load-balancer-type` 的 `external` 值是导致 AWS 负载均衡器控制器而不是 AWS 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 AWS Load Balancer Controller。有关更多信息，请参阅 AWS Load Balancer Controller 是什么？。建议升级到版本 2.7.2 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 AWS Management Console 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:1.23
          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
```

示例输出如下。

```
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
  80:32400/TCP   16h
```

> **Note**
>
> *10.100.240.137* 和 *xxxxxxxxxx-xxxxxxxxxxxxxxxx* 的值与示例输出不同（将是您的负载均衡器独有的），并且 *us-west-2* 可能因集群所在的 AWS 区域 而有所不同。

5. 打开 Amazon EC2 AWS 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 *us-west-2* 替换为在上一步输出中为 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 Linux 上的 AWS 堡垒机主机。

```
curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
```

示例输出如下。

```
<!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 AWS 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 AWS 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 AWS Network Load Balancer。有关更多信息，请参阅 Amazon EKS 上的网络负载均衡。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 AWS 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。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。
- 在集群上部署 AWS Load Balancer Controller。有关更多信息，请参阅 AWS Load Balancer Controller 是什么？。建议升级到版本 `2.7.2` 或更高版本。

- 至少两个子网位于不同的可用区。AWS 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 AWS 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 AWS 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 AWS 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 AWS 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 AWS 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 AWS CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC。
    - 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 AWS 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 AWS CloudFormation VPC 模板的更多信息，请参阅 为 Amazon EKS 集群创建 VPC。
    - 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。AWS 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，AWS 负载均衡器控制器就会创建 ALB 和必要的支持 AWS 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 AWS 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
> ```

- AWS 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 AWS 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 AWS Load Balancer Controller。有关更多信息，请参阅 AWS Load Balancer Controller 是什么？。建议升级到版本 2.7.2 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 AWS Management Console 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，AWS Load Balancer Controller 是否会创建 AWS 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/v2.7.2/docs/examples/2048/2048_full.yaml
```

   - 私有

     1. 下载清单。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-
balancer-controller/v2.7.2/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 系列创建的集群中的 Pods，请完成以下步骤。

1. 下载清单。

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.7.2/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
```

示例输出如下。

```
NAME           CLASS    HOSTS   ADDRESS
                       PORTS   AGE
```

```
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   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 AWS 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 -f -n kube-system -l app.kubernetes.io/instance=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 AWS 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/v2.7.2/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
```

示例输出如下。

```
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
```

示例输出如下。

```
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。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

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. 如果您的集群不在 us-west-2 AWS 区域，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 AWS 区域 将 *region-code* 和 *111122223333* 替换为来自 Amazon 容器镜像注册表 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||' 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。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 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 Amazon VPC CNI plugin for Kubernetes 指标帮助程序的映像。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 AWS 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 Amazon VPC CNI plugin for Kubernetes 指标帮助程序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:v1.12.6"
```

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，Amazon VPC CNI plugin for Kubernetes 指标帮助程序的 values.yaml 文件包括以下行。

```
image:
  region: us-west-2
  tag: v1.12.6
  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 中所列的每个注册表中维护映像。通过将 *602401143452* 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AWS 区域 都存在一个单独的注册表。

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   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 *602401143452* 和 *region-code* 替换为您在上一个子步骤中提供的信息。

```
docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

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 AWS 区域。

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-
code.amazonaws.com/cni-metrics-helper:v1.12.6
```

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 AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

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/cni-metrics-
helper:v1.12.6
```

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 AWS Amazon EKS 附加组件部署到集群时，您的节点会从附加组件（例如安装清单或 Helm `values.yaml` 文件）安装机制中指定的注册表中提取所需的容器映像。映像从 Amazon EKS Amazon ECR 私有存储库中提取。Amazon EKS 会将镜像复制到 AWS 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 (AWS 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 AWS IAM 账户进行身份验证。您的节点使用 Amazon EKS 节点 IAM 角色进行身份验证，该角色在与之关联的 AmazonEC2ContainerRegistryReadOnly 托管 IAM policy 中具有相关权限。

AWS 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-south-2	900889452093.dkr.ecr.ap-south-2.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
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com

AWS 区域	注册表
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
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-central-2	900612956339.dkr.ecr.eu-central-2.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-south-2	455263428931.dkr.ecr.eu-south-2.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
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com

AWS 区域	注册表
me-central-1	759879836304.dkr.ecr.me-central-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 AWS 资源进行交互。附加组件软件通常由 Kubernetes 社区、AWS 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 AWS 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 Kubernetes 字段管理。

您可以将 Amazon EKS 附加组件与任何 Amazon EKS 节点类型配合使用。

注意事项

- 要为集群配置附加组件，您的 IAM 主体必须具有 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 Kubernetes 字段管理。
- 如果您使用 AWS 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。
- eks:addon-cluster-admin ClusterRoleBinding 将 cluster-admin ClusterRole 绑定到 eks:addon-manager Kubernetes 身份。该角色拥有 eks:addon-manager 身份所需的权限，可以创建 Kubernetes 命名空间并将附加组件安装到命名空间中。如果删除 eks:addon-cluster-admin ClusterRoleBinding，Amazon EKS 集群将继续运行，但 Amazon EKS 将无法再管理任何附加组件。以下平台版本开头的所有集群都使用新的 ClusterRoleBinding。

Kubernetes 版本	EKS 平台版本
1.25	eks.2

Kubernetes 平台版本	EKS 附加组件版本
eks.14	
eks.29	
eks.35	
eks.43	

您可以使用 Amazon EKS API、AWS Management Console、AWS CLI 和 `eksctl` 添加、更新或删除 Amazon EKS 附加组件。有关更多信息，请参阅 管理 Amazon EKS 附加组件。您也可以使用 AWS CloudFormation 创建 Amazon EKS 附加组件。

Amazon EKS 提供的可用 Amazon EKS 附加组件

可在您的集群上创建以下 Amazon EKS 附加组件。您可以随时使用 `eksctl`、AWS Management Console 或 AWS CLI 查看可用附加组件的最新列表。要查看所有可用附加组件或安装附加组件，请参阅 创建附加组件。如果附加组件需要 IAM 权限，则集群必须具有 IAM OpenID Connect（OIDC）提供商。要确定是否具有一个提供商，还是创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商。安装附加组件后，您可以对其进行更新或删除。

选择一个附加组件以了解有关该附加组件及其安装要求的更多信息。

Amazon VPC CNI plugin for Kubernetes

- 名称 – `vpc-cni`
- 描述 – 为集群提供本机 VPC 联网的 Kubernetes 容器网络接口（CNI）插件。默认情况下，在每个 Amazon EC2 节点上安装自行管理的或托管类型的附加组件。
- 必需的 IAM 权限 – 此附加组件使用 Amazon EKS 的服务账户的 IAM 角色功能。如果集群使用 `IPv4` 系列，则需要 AmazonEKS_CNI_Policy 中的权限。如果集群使用 `IPv6` 系列，则必须使用 IPv6 模式中的权限创建 IAM 策略。您可以创建 IAM 角色，将其中一个策略附加到该角色，并使用以下命令为附加组件使用的 Kubernetes 服务账户添加注释。

将 *my-cluster* 替换为您的集群的名称，并将 *AmazonEKSVPCCNIRole* 替换为您的角色的名称。如果集群使用 IPv6 系列，则将 *AmazonEKS_CNI_Policy* 替换为您创建的策略名称。此命令要求您为您的设备安装 eksctl。如果您需要使用其他工具来创建角色、将策略附加到该角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

- 其他信息 – 要了解有关附加组件可配置设置的更多信息，请参阅 GitHub 上的 aws-vpc-cni-k8s。要了解有关插件的更多信息，请参阅提议：通过 AWS VPC 进行 Kubernetes 联网的 CNI 插件。有关如何创建附加组件的更多信息，请参阅 创建 Amazon EKS 附加组件。
- 更新信息 – 您一次只能更新一个次要版本。例如，如果当前版本为 1.28.*x*-eksbuild.*y*，并且您想要更新到 1.30.*x*-eksbuild.*y*，则必须首先更新到 1.29.*x*-eksbuild.*y*，再更新到 1.30.*x*-eksbuild.*y*。有关更新附加组件的更多信息，请参阅 更新 Amazon EKS 附加组件。

CoreDNS

- 名称 – coredns
- 描述 – 一个灵活、可扩展的 DNS 服务器，可用作 Kubernetes 集群 DNS。默认情况下，创建集群时会安装自行管理或托管类型的附加组件。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。如果集群包含命名空间与 CoreDNS deployment 的命名空间相匹配的 AWS Fargate 配置文件，则可以将 CoreDNS Pods 部署到 Fargate 节点。
- 所需的 IAM 权限 – 此附加组件不需要任何权限。
- 其他信息 – 要了解有关 CoreDNS 的更多信息，请参阅 Kubernetes 文档中的 Using CoreDNS for Service Discovery（使用 CoreDNS 进行服务发现）和 Customizing DNS Service（自定义 DNS 服务）。

Kube-proxy

- 名称 – kube-proxy
- 描述 – 维护每个 Amazon EC2 节点上的网络规则。它可以实现与 Pods 的网络通信。默认情况下，在集群中的每个 Amazon EC2 节点上安装自行管理或托管类型的附加组件。
- 所需的 IAM 权限 – 此附加组件不需要任何权限。

- 其他信息 – 要了解有关 kube-proxy 的更多信息，请参阅 Kubernetes 文档中的 kube-proxy。
- 更新信息 – 在更新当前版本之前，请考虑以下要求：
  - Amazon EKS 集群上的 Kube-proxy 具有与 Kubernetes 相同的兼容性和偏斜策略。
  - Kube-proxy 必须与您的Amazon EC2 节点上的 kubelet 具有相同的次要版本。
  - Kube-proxy 不能高于集群控制面板的次要版本。
  - Amazon EC2 节点上的 kube-proxy 版本不能比控制面板的版本低两个以上的次要版本。例如，如果您的控制面板正在运行 Kubernetes 1.30，则 kube-proxy 次要版本不能低于 1.28。
  - 如果您最近将集群更新到新的 Kubernetes 次要版本，请先将 Amazon EC2 节点更新到相同的次要版本，然后再将 kube-proxy 更新到与节点相同的次要版本。

Amazon EBS CSI 驱动程序

- 名称 – aws-ebs-csi-driver
- 描述 – 为集群提供 Amazon EBS 存储的 Kubernetes Container Storage Interface（CSI）插件。
- 必需的 IAM 权限 – 此附加组件使用 Amazon EKS 的服务账户的 IAM 角色功能。AmazonEBSCSIDriverPolicy AWS 托管策略中的权限是必需的。您可以使用以下命令创建 IAM 角色并将托管策略附加到其上。将 *my-cluster* 替换为您的集群的名称，并将 *AmazonEKS_EBS_CSI_DriverRole* 替换为您的角色的名称。此命令要求您为您的设备安装 eksctl。如果您需要使用其他工具或需要使用自定义 KMS 密钥进行加密，请参阅 创建 Amazon EBS CSI 驱动程序 IAM 角色。

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
    --approve
```

- 其他信息 – 要了解有关附加组件的更多信息，请参阅 Amazon EBS CSI 驱动程序。

Amazon EFS CSI 驱动程序

- 名称 – aws-efs-csi-driver

- 描述 – 为集群提供 Amazon EFS 存储的 Kubernetes 容器存储接口（CSI）插件。
- 必需的 IAM 权限 – 此附加组件使用 Amazon EKS 的服务账户的 IAM 角色功能。AmazonEFSCSIDriverPolicy AWS 托管策略中的权限是必需的。您可以使用以下命令创建 IAM 角色并为其附加托管式策略。将 *my-cluster* 替换为您的集群的名称，并将 *AmazonEKS_EFS_CSI_DriverRole* 替换为您的角色的名称。这些命令要求您的设备事先安装 eksctl。如果您需要使用其他工具，请参阅 创建 IAM 角色。

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

- 其他信息 – 要了解有关附加组件的更多信息，请参阅 Amazon EFS CSI 驱动程序。

适用于 Amazon S3 的 Mountpoint CSI 驱动程序

- 名称 – aws-mountpoint-s3-csi-driver
- 描述：一个 Kubernetes 容器存储接口（CSI）插件，可为集群提供 Amazon S3 存储。
- 必需的 IAM 权限 – 此附加组件使用 Amazon EKS 的服务账户的 IAM 角色功能。创建的 IAM 角色需要一个允许访问 S3 的策略。创建策略时，请遵循 Mountpoint IAM 权限建议。或者，您可以使用 AWS 托管策略 AmazonS3FullAccess，但此托管策略授予的权限超出了 Mountpoint 所需的权限。

  您可以使用以下命令创建 IAM 角色，并为其附加策略。将 *my-cluster* 替换为集群的名称，将 *region-code* 替换为正确的 AWS 区域 代码，将 *AmazonEKS_S3_CSI_DriverRole* 替换为角色的名称，将 *AmazonEKS_S3_CSI_DriverRole_ARN* 替换为角色 ARN。这些命令要求您的设备事先安装 eksctl。有关如何使用 IAM 控制台或 AWS CLI 的说明，请参阅 创建 IAM 角色。

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

- 其他信息 – 要了解有关附加组件的更多信息，请参阅 适用于 Amazon S3 的 Mountpoint CSI 驱动程序。

CSI 快照控制器

- 名称 – `snapshot-controller`
- 描述：容器存储接口（CSI）快照控制器允许在兼容的 CSI 驱动程序（例如 Amazon EBS CSI 驱动程序）中使用快照功能。
- 所需的 IAM 权限 – 此附加组件不需要任何权限。
- 其他信息 – 要了解有关附加组件的更多信息，请参阅 CSI 快照控制器。

适用于 OpenTelemetry 的 AWS Distro

- 名称 – `adot`
- 描述 – AWS Distro for OpenTelemetry（ADOT）是 OpenTelemetry 项目的安全、受 AWS 支持的生产就绪型发行版。
- 所需的 IAM 权限：只有在使用可通过高级配置选择加入的预配置自定义资源时，该插件才需要 IAM 权限。
- 其他信息：有关更多信息，请参阅适用于 OpenTelemetry 的 AWS Distro 文档中的使用 EKS 插件且适用于 OpenTelemetry 的 AWS Distro 入门。

  ADOT 要求将 `cert-manager` 作为先决条件部署在集群上，否则如果使用 Amazon EKS Terraform `cluster_addons` 属性直接部署，此插件将无法运行。有关更多要求，请参阅适用于

OpenTelemetry 的 AWS Distro 文档中的使用 EKS 插件且适用于 OpenTelemetry 的 AWS Distro 入门要求。

Amazon GuardDuty 代理

- 名称 – `aws-guardduty-agent`
- 描述 — Amazon GuardDuty 是一项安全监控服务，用于分析和处理基础数据源，包括 AWS CloudTrail 管理事件和 Amazon VPC 流日志。Amazon GuardDuty 还处理 Kubernetes 审计日志和运行时监控等功能。
- 所需的 IAM 权限 – 此附加组件不需要任何权限。
- 其他信息 – 有关更多信息，请参阅 Amazon GuardDuty 中的 Amazon EKS 集群的运行时监控。
  - 要检测 Amazon EKS 集群中的潜在安全威胁，请启用 Amazon GuardDuty 运行时监控，然后将 GuardDuty 安全代理部署到您的 Amazon EKS 集群。

Amazon CloudWatch 可观测性代理

- 名称 – `amazon-cloudwatch-observability`
- 描述：Amazon CloudWatch 代理 是 AWS 提供的监控和可观测性服务。此插件将安装 CloudWatch 代理，并启用 CloudWatch Application Signals 和 CloudWatch Container Insights，从而增强 Amazon EKS 的可观测性。
- 必需的 IAM 权限 – 此附加组件使用 Amazon EKS 的服务账户的 IAM 角色功能。AWSXrayWriteOnlyAccess 和 CloudWatchAgentServerPolicy AWS 托管策略中的权限是必需的。您可以创建 IAM 角色，将托管策略附加到该角色，并使用以下命令为附加组件使用的 Kubernetes 服务账户添加注释。将 *my-cluster* 替换为您的集群的名称，并将 *AmazonEKS_Observability_role* 替换为您的角色的名称。此命令要求您为您的设备安装 `eksctl`。如果您需要使用其他工具来创建角色、将策略附加到该角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
```

```
    --approve
```

- 其他信息：有关更多信息，请参阅安装 CloudWatch 代理。

Amazon EKS 容器组身份代理

- 名称 – `eks-pod-identity-agent`
- 描述：Amazon EKS 容器组身份提供管理应用程序凭证的功能，类似于 Amazon EC2 实例配置文件为 EC2 实例提供凭证的方式。
- 所需的 IAM 权限：此插件的用户权限来自 Amazon EKS 节点 IAM 角色。
- 更新信息 – 您一次只能更新一个次要版本。例如，如果当前版本为 `1.28.`*x*`-eksbuild.`*y*，并且您想要更新到 `1.30.`*x*`-eksbuild.`*y*，则必须首先更新到 `1.29.`*x*`-eksbuild.`*y*，再更新到 `1.30.`*x*`-eksbuild.`*y*。有关更新附加组件的更多信息，请参阅 更新 Amazon EKS 附加组件。

来自独立软件供应商的其他 Amazon EKS 附加组件

除了之前的 Amazon EKS 附加组件列表外，您还可以添加来自独立软件供应商的各种操作软件 Amazon EKS 附加组件。选择一个附加组件以了解有关该附加组件及其安装要求的更多信息。

从 AWS Marketplace 查找、购买插件并将其部署到 Amazon EKS（YouTube）。

Accuknox

- 发布者：Accuknox
- 名称 – `accuknox_kubearmor`
- 命名空间：`kubearmor`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 KubeArmor 文档中的 KubeArmor 入门。

Akuity

- 发布者：Akuity
- 名称 – `akuity_agent`

- 命名空间：akuity
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Akuity 平台文档中的 Installing the Akuity Agent on Amazon EKS with the Akuity EKS add-on。

Calyptia

- 发布者：Calyptia
- 名称 – calyptia_fluent-bit
- 命名空间：calytia-fluentbit
- 服务账户名称：clyptia-fluentbit
- AWS 托管 IAM 策略：AWSMarketplaceMeteringRegisterUsage。
- 创建所需 IAM 角色的命令：以下命令要求您的集群具有 IAM OpenID Connect（OIDC）提供商。要确定是否具有一个提供商，还是创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商。将 *my-cluster* 替换为您的集群的名称，并将 *my-calyptia-role* 替换为您的角色的名称。此命令要求您为您的设备安装 eksctl。如果您需要使用其他工具来创建角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

- 设置和使用说明：请参阅 Calyptia 文档中的 Calyptia for Fluent Bit。

Cisco Observability Collector

- 发布者：Cisco
- 名称 – cisco_cisco-cloud-observability-collectors
- 命名空间：appdynamics
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。

- 设置和使用说明 – 请参阅 Cisco AppDynamics 文档中的 Use the Cisco Cloud Observability AWS Marketplace Add-Ons。

Cisco Observability Operator

- 发布者：Cisco
- 名称 – `cisco_cisco-cloud-observability-operators`
- 命名空间：`appdynamics`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Cisco AppDynamics 文档中的 Use the Cisco Cloud Observability AWS Marketplace Add-Ons。

CLOUDSOFT

- 发布者：CLOUDSOFT
- 名称 – `cloudsoft_cloudsoft-amp`
- 命名空间：`cloudsoft-amp`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 CLOUDSOFT 文档中的 Amazon EKS ADDON。

Cribl

- 发布者：Cribl
- 名称 – `cribl_cribledge`
- 命名空间：`cribledge`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Cribl 文档中的安装适用于边缘的 Cribl Amazon EKS 插件。

Dynatrace

- 发布者：Dynatrace
- 名称 – dynatrace_dynatrace-operator
- 命名空间：dynatrace
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 dynatrace 文档中的 Kubernetes monitoring（Kubernetes 监控）。

Datree

- 发布者：Datree
- 名称 – datree_engine-pro
- 命名空间：datree
- 服务账户名称 – datree-webhook-server-awsmp
- AWS 托管 IAM 策略：AWSLicenseManagerConsumptionPolicy。
- 创建所需 IAM 角色的命令：以下命令要求您的集群具有 IAM OpenID Connect（OIDC）提供商。要确定是否具有一个提供商，还是创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商。将 *my-cluster* 替换为您的集群的名称，并将 *my-datree-role* 替换为您的角色的名称。此命令要求您为您的设备安装 eksctl。如果您需要使用其他工具来创建角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

  ```
  eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
   --cluster my-cluster --role-name my-datree-role \
      --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
  AWSLicenseManagerConsumptionPolicy --approve
  ```

- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Datree 文档中的 Amazon EKS-集成。

Datadog

- 发布者：Datadog
- 名称 – datadog_operator

- 命名空间：`datadog-agent`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Datadog 文档中的使用 Datadog Operator 插件在 Amazon EKS 上安装 Datadog 代理。

Groundcover

- 发布者：groundcover
- 名称 – `groundcover_agent`
- 命名空间：`groundcover`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 groundcover 文档中的安装 groundcover Amazon EKS 插件。

Grafana Labs

- 发布者：Grafana Labs
- 名称 – `grafana-labs_kubernetes-monitoring`
- 命名空间：`monitoring`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Grafana Labs 文档中的 Configure Kubernetes Monitoring as an Add-on with Amazon EKS。

HA Proxy

- 发布者：HA Proxy
- 名称 – `haproxy-technologies_kubernetes-ingress-ee`

- 命名空间：haproxy-controller
- 服务账户名称：customer defined
- AWS 托管 IAM 策略：AWSLicenseManagerConsumptionPolicy。
- 创建所需 IAM 角色的命令：以下命令要求您的集群具有 IAM OpenID Connect（OIDC）提供商。要确定是否具有一个提供商，还是创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商。将 *my-cluster* 替换为您的集群的名称，并将 *my-haproxy-role* 替换为您的角色的名称。此命令要求您为您的设备安装 eksctl。如果您需要使用其他工具来创建角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 HAProxy 文档中的 从 AWS 在 Amazon EKS 上安装 HAProxy Enterprise Kubernetes 入口控制器。

Kpow

- 发布者：Factorhouse
- 名称 – factorhouse_kpow
- 命名空间：factorhouse
- 服务账户名称：kpow
- AWS 托管 IAM 策略：AWSLicenseManagerConsumptionPolicy
- 创建所需 IAM 角色的命令：以下命令要求您的集群具有 IAM OpenID Connect（OIDC）提供商。要确定是否具有一个提供商，还是创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商。将 *my-cluster* 替换为您的集群的名称，并将 *my-kpow-role* 替换为您的角色的名称。此命令要求您为您的设备安装 eksctl。如果您需要使用其他工具来创建角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- 自定义 IAM 权限：此附加组件不使用自定义权限。

- 设置和使用说明：请参阅 Kpow 文档中的 AWS Marketplace LM。

Kubecost

- 发布者：Kubecost
- 名称 – `kubecost_kubecost`
- 命名空间：`kubecost`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Kubecost 文档中的 AWS 云账单集成。
- 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”，否则您将收到错误信息。

Kasten

- 发布者：Kasten by Veeam
- 名称 – `kasten_k10`
- 命名空间：`kasten-io`
- 服务账户名称：`k10-k10`
- AWS 托管 IAM 策略：AWSLicenseManagerConsumptionPolicy。
- 创建所需 IAM 角色的命令：以下命令要求您的集群具有 IAM OpenID Connect（OIDC）提供商。要确定是否具有一个提供商，还是创建一个提供商，请参阅 为集群创建 IAM OIDC 提供商。将 *my-cluster* 替换为您的集群的名称，并将 *my-kasten-role* 替换为您的角色的名称。此命令要求您为您的设备安装 `eksctl`。如果您需要使用其他工具来创建角色并为 Kubernetes 服务账户添加注释，请参阅 配置 Kubernetes 服务账户来代入 IAM 角色。

  ```
  eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-cluster --role-name my-kasten-role \
      --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/AWSLicenseManagerConsumptionPolicy --approve
  ```

- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Kasten 文档中的使用 Amazon EKS 附加组件在 AWS 上安装 K10。

- 其他信息 – 如果您的 Amazon EKS 集群为版本 Kubernetes `1.23` 或更高版本，则必须使用默认的 `StorageClass` 在集群上安装 Amazon EBS CSI 驱动程序。

Kong

- 发布者：Kong
- 名称 – `kong_konnect-ri`
- 命名空间：`kong`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Kong 文档中的安装 Kong Gateway EKS 插件。

LeakSignal

- 发布者：LeakSignal
- 名称 – `leaksignal_leakagent`
- 命名空间：`leakagent`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 LeakSignal 文档中的 Install the LeakAgent add-on。

NetApp

- 发布者：NetApp
- 名称 – `netapp_trident-operator`
- 命名空间：`trident`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 - 请参阅 NetApp 文档中的配置 Astra Trident EKS 附加组件。

New Relic

- 发布者：New Relic
- 名称 – `new-relic_kubernetes-operator`
- 命名空间：`newrelic`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 New Relic 文档中的安装适用于 EKS 的 New Relic 插件。

Rafay

- 发布者：Rafay
- 名称 – `rafay-systems_rafay-operator`
- 命名空间：`rafay-system`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Rafay 文档中的安装 Rafay Amazon EKS 插件。

Solo.io

- 发布者：Solo.io
- 名称 – `solo-io_istio-distro`
- 命名空间：`istio-system`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 — 请参阅 Solo.io 文档中的安装 Istio。

Stormforge

- 发布者：Stormforge

- 名称 – `stormforge_optimize-Live`
- 命名空间：`stormforge-system`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 StormForge 文档中的安装 StormForge 代理。

Splunk

- 发布者：Splunk
- 名称 – `splunk_splunk-otel-collector-chart`
- 命名空间：`splunk-monitoring`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Splunk 文档中的 Install the Splunk add-on for Amazon EKS。

Teleport

- 发布者：Teleport
- 名称 – `teleport_teleport`
- 命名空间：`teleport`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Teleport 文档中的 How Teleport Works（Teleport 的工作原理）。

Tetrate

- 发布者 – Tetrate Io
- 名称 – `tetrate-io_istio-distro`
- 命名空间：`istio-system`

- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Tetrate Istio Distro 网站。

Upbound Universal Crossplane

- 发布者：Upbound
- 名称 – `upbound_universal-crossplane`
- 命名空间：`upbound-system`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明：请参阅 Upbound 文档中的 Upbound Universal Crossplane（UXP）。

Upwind

- 发布者：Upwind
- 名称 – `upwind`
- 命名空间：`upwind`
- 服务账户名称：此附加组件未使用服务账户。
- AWS 托管 IAM 策略：此附加组件不使用托管策略。
- 自定义 IAM 权限：此附加组件不使用自定义权限。
- 设置和使用说明 – 请参阅 Upwind 文档中的安装步骤。

管理 Amazon EKS 附加组件

Amazon EKS 附加组件是一组用于 Amazon EKS 集群的精选附加软件。所有 Amazon EKS 附加组件都具有以下特性：

- 包含最新的安全补丁和错误修复。
- 经过 AWS 验证，可与 Amazon EKS 一起使用。

- 可减少管理附加软件所需的工作量。

当 Amazon EKS 附加组件有新版本时，AWS Management Console会通知您。您只需启动更新，Amazon EKS 就会为您更新附加软件。

有关可用附加组件的列表，请参阅Amazon EKS 提供的可用 Amazon EKS 附加组件。有关 Kubernetes 字段管理的更多信息，请参阅 Kubernetes 字段管理。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。

创建附加组件

您可以使用 `eksctl`、AWS Management Console或 AWS CLI 创建 Amazon EKS 附加组件。如果附加组件需要 IAM 角色，请参阅 Amazon EKS 提供的可用 Amazon EKS 附加组件 中特定附加组件的详细信息，以了解有关创建角色的详细信息。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

使用 **`eksctl`** 创建 Amazon EKS 附加组件

1. 查看某个集群版本的可用附加组件名称。将 *`1.30`* 替换为您的集群版本。

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 | grep AddonName
```

示例输出如下。

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
                        "AddonName": "upbound_universal-crossplane",
                        "AddonName": "teleport_teleport",
```

```
                    "AddonName": "factorhouse_kpow",
                    [...]
```

2. 查看要创建的附加组件的可用版本。将 *1.30* 替换为您的集群版本。将 *name-of-addon* 替换为您要查看其版本的附加组件的名称。该名称必须是前面步骤中返回的名称之一。

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 --name name-of-addon | grep AddonVersion
```

以下输出是一个示例，显示了为名为 vpc-cni 的附加组件返回的内容。您可以看到该附加组件有多个可用版本。

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

3. 确定您要创建的附加组件是 Amazon EKS 还是 AWS Marketplace 附加组件。AWS Marketplace 具有第三方附加组件，需要您完成额外的步骤才能创建附加组件。

```
eksctl utils describe-addon-versions --kubernetes-version 1.30 --name name-of-addon | grep ProductUrl
```

如果未返回输出，则该附加组件是 Amazon EKS。如果返回输出，则该附加组件是 AWS Marketplace 附加组件。以下输出适用于名为 teleport_teleport 的附加组件。

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?
sku=3bda70bb-566f-4976-806c-f96faef18b26"
```

您可以通过返回的 URL 在 AWS Marketplace 中了解有关该附加组件的更多信息。
如果该附加组件需要订阅，您可以通过 AWS Marketplace 对其进行订阅。如果您要
从 AWS Marketplace 创建附加组件，则用于创建附加组件的 IAM 主体必须具有创建
AWSServiceRoleForAWSLicenseManagerRole 服务相关角色的权限。有关向 IAM 实体分配权限的更多信息，请参阅《IAM 用户指南》中的添加和移除 IAM 身份权限。

4. 创建 Amazon EKS 附加组件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令：

   - 将 *my-cluster* 替换为您的集群名称。

- 将 *name-of-addon* 替换为您要创建的附加组件的名称。
- 如果您需要早于最新版本的附加组件版本，请将 *latest* 替换为前面步骤的输出中返回的适用版本号。
- 如果附加组件使用了服务账户角色，请将 *111122223333* 替换为您的账户 ID，并将 *role-name* 替换为该角色的名称。有关为服务账户创建角色的说明，请参阅您正在创建的附加组件的文档。指定服务账户角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅为集群创建 IAM OIDC 提供商。

  如果附加组件不使用服务账户角色，请删除 ***--service-account-role-arn arn:aws:iam::111122223333:role/role-name***。
- 此示例命令将覆盖附加组件的任何现有自行管理版本（如果有的话）的配置。如果您不想覆盖现有的自行管理附加组件的配置，请删除 *--force* 选项。如果您删除此选项，并且 Amazon EKS 附加组件需要覆盖现有的自行管理附加组件的配置，那么创建 Amazon EKS 附加组件将会失败，并显示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。

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest
 \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
force
```

您可以看到该命令所有可用选项的列表。

```
eksctl create addon --help
```

有关可用选项的更多信息，请参阅 `eksctl` 文档中的 Addons（附加组件）。

AWS Management Console

使用 AWS Management Console 创建 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），然后选择要为其创建附加组件的集群的名称。
3. 选择附加组件选项卡。
4. 选择获取更多附加组件。

5. 选择要添加到集群的附加组件。您可以根据需要添加任意数量的 Amazon EKS 附加组件和 AWS Marketplace 附加组件。

   对于 AWS Marketplace 附加组件，用于创建附加组件的 IAM 主体必须有权从 AWS LicenseManager 读取附加组件的权限。AWSLicenseManager 需要 AWSServiceRoleForAWSLicenseManagerRole 服务相关角色（SLR），该角色允许 AWS 资源代表您管理许可证。SLR 是每个账户的一次性要求，您无需为每个附加组件或每个集群创建单独的 SLR。有关向 IAM 主体分配权限的更多信息，请参阅《IAM 用户指南》中的添加和移除 IAM 身份权限。

   如果未列出要安装的 AWS Marketplace 附加组件，则可以通过在搜索框中输入文本来搜索可用的附加组件。在筛选选项中，您也可以按类别、供应商或定价模式进行搜索，然后从搜索结果中选择附加组件。选择要安装的附加组件后，选择 Next（下一步）。

6. 在 Configure selected add-ons settings（配置选定的附加组件设置）页面上：

   - 选择查看订阅选项，打开订阅选项表单。查看定价详细信息和法律部分，然后选择订阅按钮继续。
   - 对于 Version（版本），请选择要安装的版本。我们建议选择标记为 latest（最新）的版本，除非您正在创建的个别附加组件推荐了不同的版本。要确定附加组件是否有推荐版本，请参阅您正在创建的附加组件的文档。
   - 如果您选择的所有附加组件在 Status（状态）下都有 Requires subscription（需要订阅）的字样，请选择 Next（下一步）。创建集群后，您在订阅这些附加组件之前无法进一步配置此类附加组件。对于 Status（状态）下没有 Requires subscription（需要订阅）字样的附加组件：
     - 对于 Select IAM role（选择 IAM 角色），接受默认选项，除非该附加组件需要 IAM 权限。如果该附加组件需要 AWS 权限，则可以选择节点的 IAM 角色（未设置）或为该附加组件创建的现有角色。如果没有可选择的角色，则表示没有现有角色。无论您选择哪个选项，请参阅您正在创建的附加组件（用于创建IAM 策略并将其附加到某个角色）的文档。选择 IAM 角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅为集群创建 IAM OIDC 提供商。
     - 选择 Optional configuration settings（可选配置设置）。
       - 如果附加组件需要配置，请在 Configuration values（配置值）框中输入相应的值。要确定附加组件是否需要配置信息，请参阅您正在创建的附加组件的文档。
       - 为 Conflict resolution method（冲突解决方法）选择一个可用的选项。
     - 选择下一步。

7. 在查看和添加页面上，选择创建。附加组件安装完成后，您会看到您安装的附加组件。
8. 如果您安装的任何附加组件需要订阅，请完成以下步骤：

   1. 选择该附加组件右下角的 Subscribe（订阅）按钮。您将转到该附加组件在 AWS Marketplace 中的页面。阅读有关该附加组件的信息，例如其 Product Overview（产品概述）和 Pricing Information（定价信息）。
   2. 选择附加组件页面右上角的 Continue to Subscribe（继续订阅）按钮。
   3. 通读 Terms and Conditions（条款和条件）。如果您同意这些条款和条件，请选择 Accept Terms（接受条款）。处理订阅可能需要几分钟的时间。在处理订阅时，Return to Amazon EKS Console（返回 Amazon EKS 控制台）按钮将显示为灰色。
   4. 订阅处理完成后，Return to Amazon EKS Console（返回 Amazon EKS 控制台）按钮将不再显示为灰色。选择此按钮返回集群的 Amazon EKS 控制台 Add-ons（附加组件）选项卡。
   5. 对于您已订阅的附加组件，请选择 Remove and reinstall（删除并重新安装），然后选择 Reinstall add-on（重新安装附加组件）。安装该附加组件可能需要几分钟的时间。安装完成后，您可以配置该附加组件。

AWS CLI

先决条件

在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

使用 AWS CLI 创建 Amazon EKS 附加组件

1. 确定哪些附加组件可用。您可以看到所有可用的附加组件及其类型和发布者。您还可以看到通过 AWS Marketplace 提供的附加组件的 URL。将 *`1.30`* 替换为您的集群版本。

```
aws eks describe-addon-versions --kubernetes-version 1.30 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

示例输出如下。

```
-------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
       |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
|                     MarketplaceProductUrl                     |
 Name               |      Owner       |  Publisher   |        Type         |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
|  None                                                         |  aws-ebs-csi-
driver           |  aws             |  eks         |  storage            |
|  None                                                         |  coredns
                |  aws             |  eks         |  networking         |
|  None                                                         |  kube-proxy
                |  aws             |  eks         |  networking         |
|  None                                                         |  vpc-cni
                |  aws             |  eks         |  networking         |
|  None                                                         |  adot
                |  aws             |  eks         |  observability      |
|  https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
      |
|  https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |
 upbound_universal-crossplane |  aws-marketplace |  upbound     |  infra-
management   |
|  https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |
 teleport_teleport            |  aws-marketplace |  teleport    |  policy-
management  |
|  https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |
 factorhouse_kpow             |  aws-marketplace |  factorhouse |  monitoring
      |
|  [...]                                                        |  [...]
                |  [...]           |  [...]       |  [...]              |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
```

您的输出可能会有所不同。在此输出示例中，有三个不同的 networking 类型的附加组件和五个发布者类型为 eks 的附加组件。在 Owner 列中值为 aws-marketplace 的附加组件可能需要订阅才能安装。您可以访问相应的 URL 以了解有关该附加组件的更多信息并进行订阅。

2. 您可以看到每个附加组件的可用版本。将 *1.30* 替换为您的集群版本，将 *vpc-cni* 替换为上一步返回的附加组件名称。

```
aws eks describe-addon-versions --kubernetes-version 1.30 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

示例输出如下。

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

默认情况下，Defaultversion 列中值为 True 的版本是创建附加组件时使用的版本。

3. （可选）通过运行以下命令查找所选附加组件的配置选项：

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-
version v1.12.0-eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema
\":\"http://json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":
{\"additionalProperties\":false,\"properties\":{\"hostPath\":{\"$ref\":
\"#/definitions/HostPath\"}},\"title\":\"Cri\",\"type\":\"object\"},\"Env
\":{\"additionalProperties\":false,\"properties\":{\"ADDITIONAL_ENI_TAGS
\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":\"integer
```

```
\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":
{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type
\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT\":{\"type\":
\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string
\"},\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION
\":{\"format\":\"boolean\",\"type\":\"string\"},\"DISABLE_METRICS\":{\"format
\":\"boolean\",\"type\":\"string\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING
\":{\"format\":\"boolean\",\"type\":\"string\"},\"ENABLE_POD_ENI\":{\"format
\":\"boolean\",\"type\":\"string\"},\"ENABLE_PREFIX_DELEGATION\":{\"format
\":\"boolean\",\"type\":\"string\"},\"WARM_ENI_TARGET\":{\"format\":\"integer
\",\"type\":\"string\"},\"WARM_PREFIX_TARGET\":{\"format\":\"integer\",
\"type\":\"string\"}},\"title\":\"Env\",\"type\":\"object\"},\"HostPath\":
{\"additionalProperties\":false,\"properties\":{\"path\":{\"type\":\"string\"}},
\"title\":\"HostPath\",\"type\":\"object\"},\"Limits\":{\"additionalProperties
\":false,\"properties\":{\"cpu\":{\"type\":\"string\"},\"memory\":{\"type
\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},\"Resources\":
{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":\"#/
definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},
\"title\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties
\":false,\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":
{\"$ref\":\"#/definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/
Resources\"}},\"title\":\"VpcCni\",\"type\":\"object\"}}}"
}
```

输出是标准的 JSON 架构。

以下是适用于上述架构的 JSON 格式的有效配置值示例。

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

以下是适用于上述架构的 YAML 格式的有效配置值示例。

```
resources:
  limits:
    cpu: 100m
```

4. 确定该附加组件是否需要 IAM 权限。如果是，则需要（1）确定是否要使用 EKS 容器组身份或服务账户的 IAM 角色（IRSA），（2）确定要与附加组件一起使用的 IAM 角色的 ARN，以及（3）确定附加组件使用的 Kubernetes 服务账户的名称。您可以在文档中或使用 AWS API 找到此信息，请参阅检索有关附加组件的 IAM 信息。

   - 如果附加组件支持，Amazon EKS 建议使用 EKS 容器组身份。这需要在您的集群上安装容器组身份代理。有关将容器组身份与附加组件配合使用的更多信息，请参阅 使用容器组身份将 IAM 角色附加到 Amazon EKS 附加组件。
   - 如果附加组件或您的集群未针对 EKS 容器组身份进行设置，请使用 IRSA。确认您的集群上已设置 IRSA。
   - 审查 Amazon EKS 附加组件文档，确定该附加组件是否需要 IAM 权限以及关联 Kubernetes 服务账户的名称。
5. 创建 Amazon EKS 附加组件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令：

   - 将 *my-cluster* 替换为您的集群名称。
   - 将 *vpc-cni* 替换为前面步骤的输出中返回的要创建的附加组件名称。
   - 将 *version-number* 替换为前面步骤的输出中返回的要使用的版本。
   - 如果附加组件不需要 IAM 权限，则删除 *<service-account-configuration>*。
   - 如果附加组件（1）需要 IAM 权限，并且（2）您的集群使用 EKS 容器组身份，请将 *<service-account-configuration>* 替换为以下容器组身份关联。将 *<service-account-name>* 替换为附加组件使用的服务账户名称。将 *<role-arn>* 替换为 IAM 角色的 ARN。该角色必须拥有 EKS 容器组身份所需的信任策略。

     ```
     --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
     ```

   - 如果附加组件（1）需要 IAM 权限，并且（2）您的集群使用 IRSA，请将 *<service-account-configuration>* 替换为以下 IRSA 配置。将 *111122223333* 替换为您的账户 ID，并将 *role-name* 替换为您创建的现有 IAM 角色的名称。有关创建角色的说明，请参阅您正在创建的附加组件的文档。指定服务账户角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅 为集群创建 IAM OIDC 提供商。

  - `--service-account-role-arn arn:aws:iam::111122223333:role/role-name`
- 这些示例命令将覆盖附加组件的任何现有自行管理版本（如果有的话）的 `--configuration-values` 选项。将其替换为所需配置值，例如字符串或文件输入。如果不想提供配置值，请删除 `--configuration-values` 选项。如果您不希望 AWS CLI 覆盖现有的自行管理附加组件的配置，请删除 *`--resolve-conflicts OVERWRITE`* 选项。如果您删除此选项，并且 Amazon EKS 附加组件需要覆盖现有的自行管理附加组件的配置，那么创建 Amazon EKS 附加组件将会失败，并显示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。

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
     <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    <service-account-configuration> --configuration-values 'file://example.yaml'
 --resolve-conflicts OVERWRITE
```

有关可用选项的完整列表，请参阅 Amazon EKS Command Line Reference（《Amazon EKS 命令行参考》）中的 create-addon。如果您创建的附加组件在前面步骤的 Owner 列中的值为 aws-marketplace，那么创建可能会失败，您可能会收到与以下错误类似的错误消息。

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-
on. To subscribe, visit the AWS Marketplace console, agree to the seller EULA,
 select the pricing type if required, then re-install the add-on"
```

```
[...]
```

如果您收到的错误与之前输出中的错误类似，请访问前面步骤的输出中的 URL，以订阅附加组件。订阅后，再次运行 `create-addon` 命令。

更新附加组件

在新版本发布时或您将集群更新到新的 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，必须启动更新。启动更新后，Amazon EKS 会为您更新附加组件。在更新某个附加组件之前，请查看该附加组件的最新文档。有关可用附加组件的列表，请参阅 Amazon EKS 提供的可用 Amazon EKS 附加组件。如果附加组件需要 IAM 角色，请参阅 Amazon EKS 提供的可用 Amazon EKS 附加组件 中特定附加组件的详细信息，以了解有关创建角色的详细信息。

您可以使用 `eksctl`、AWS Management Console 或 AWS CLI 更新 Amazon EKS 附加组件。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

使用 **`eksctl`** 更新 Amazon EKS 附加组件

1. 确定集群上目前安装的附加组件和附加组件版本。将 *`my-cluster`* 替换为您的集群名称。

```
eksctl get addon --cluster my-cluster
```

示例输出如下。

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                v1.12.0-
eksbuild.1,v1.11.4-eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-
eksbuild.1
```

您的输出可能会有所不同，这取决于您的集群上安装的附加组件和版本。您可以看到，在前面的输出示例中，集群上的两个现有附加组件在 `UPDATE AVAILABLE` 列中有较新的版本可用。

2. 更新附加组件。

1. 将以下命令复制到您的设备。根据需要对该命令进行以下修改：
   - 将 *my-cluster* 替换为您的集群名称。
   - 请将 *region-code* 替换为集群所在的 AWS 区域。
   - 将 *vpc-cni* 替换为上一步的输出中返回的要更新的附加组件名称。
   - 如果要更新到的版本早于最新版本，请将 *latest* 替换为上一步的输出中返回的适用版本号。一些附加组件有推荐版本。有关更多信息，请参阅您正在更新的附加组件的文档。
   - 如果附加组件使用 Kubernetes 服务账户和 IAM 角色，请将 *111122223333* 替换为您的账户 ID，并将 *role-name* 替换为您创建的现有 IAM 角色的名称。有关创建角色的说明，请参阅您正在创建的附加组件的文档。指定服务账户角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅 为集群创建 IAM OIDC 提供商。

     如果附加组件不使用 Kubernetes 服务账户和 IAM 角色，请删除 **serviceAccountRoleARN: arn:aws:iam::*111122223333*:role/*role-name*** 行。
   - *preserve* 选项保留附加组件的现有值。如果您为附加组件设置设定了自定义值，但未使用此选项，则 Amazon EKS 会使用其默认值覆盖您的值。如果您使用此选项，那么我们建议您在更新生产集群上的附加组件之前，先测试非生产集群上所有更改的字段和值。如果您将该值改为 overwrite，则所有设置都将更改为 Amazon EKS 的默认值。如果您为任何设置设定了自定义值，这些值可能会被 Amazon EKS 的默认值覆盖。如果您将该值改为 none，Amazon EKS 不会更改任何设置的值，但更新可能会失败。如果更新失败，您会收到一条帮助您解决冲突的错误消息。

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

2. 运行修改后的命令以创建 `update-addon.yaml` 文件。
3. 将配置文件应用到您的集群。

```
eksctl update addon -f update-addon.yaml
```

有关更新附加组件的更多信息，请参阅 `eksctl` 文档中的 Addons（附加组件）。

AWS Management Console

使用 AWS Management Console 更新 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附加组件的集群的名称。
3. 选择附加组件选项卡。
4. 选择附加组件框右上角的框，然后选择 Edit（编辑）。
5. 在 Configure ***name of addon***（配置 name of addon）页面上：
   - 选择您想使用的 Version（版本）。该附加组件可能有推荐版本。有关更多信息，请参阅您正在更新的附加组件的文档。
   - 对于选择 IAM 角色，您可以使用节点的 IAM 角色（未设t）或为该附加组件创建的现有角色。如果没有可选择的角色，则表示没有现有角色。无论您选择哪个选项，请参阅您正在创建的附加组件（用于创建IAM 策略并将其附加到某个角色）的文档。选择 IAM 角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅为集群创建 IAM OIDC 提供商。
   - 对于 `Code editor`，请输入附加组件特有的任何配置信息。有关更多信息，请参阅您正在更新的附加组件的文档。
   - 对于 Conflict resolution method（冲突解决方法），选择其中一个选项。如果您为附加组件设置设定了自定义值，我们建议选择 Preserve（保留）选项。如果您不选择此选项，Amazon EKS 会使用其默认值覆盖您的值。如果您使用此选项，那么我们建议您在更新生产集群上的附加组件之前，先测试非生产集群上所有更改的字段和值。
6. 选择更新。

AWS CLI

先决条件

在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 2.12.3 或更高版本，或版本 1.27.160 或更高版本。要查看当前版本，请使用 **aws --version | cut -d / -f2 | cut -d ' ' -f1**。软件包管理器（如 yum、apt-get 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。

使用 AWS CLI 更新 Amazon EKS 附加组件

1. 查看已安装的附加组件列表。将 *my-cluster* 替换为您的集群名称。

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

示例输出如下。

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni"
    ]
}
```

2. 查看要更新的附加组件的当前版本。将 *my-cluster* 替换为您的集群名称，将 *vpc-cni* 替换为要更新的附加组件的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

示例输出如下。

```
v1.10.4-eksbuild.1
```

3. 您可以看到该附加组件的哪些版本可用于您的集群版本。将 *1.30* 替换为您的集群版本，将 *vpc-cni* 替换为要更新的附加组件的名称。

```
aws eks describe-addon-versions --kubernetes-version 1.30 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

示例输出如下。

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

默认情况下，Defaultversion 列中值为 True 的版本是创建附加组件时使用的版本。

4. 更新您的附加组件。将以下命令复制到您的设备。根据需要对该命令进行以下修改，然后运行修改后的命令。

   - 将 *my-cluster* 替换为您的集群名称。
   - 将 *vpc-cni* 替换为前面步骤的输出中返回的要更新的附加组件名称。
   - 将 *version-number* 替换为上一步的输出中返回的要更新到的版本。一些附加组件有推荐版本。有关更多信息，请参阅您正在更新的附加组件的文档。
   - 如果附加组件使用 Kubernetes 服务账户和 IAM 角色，请将 *111122223333* 替换为您的账户 ID，并将 *role-name* 替换为您创建的现有 IAM 角色的名称。有关创建角色的说明，请参阅您正在创建的附加组件的文档。指定服务账户角色需要您的集群具有 IAM OpenID Connect（OIDC）提供程序。要确定您的集群是否具有此提供程序，或者要创建此提供程序，请参阅 为集群创建 IAM OIDC 提供商。

     如果附加组件不使用 Kubernetes 服务账户和 IAM 角色，请删除 **serviceAccountRoleARN: arn:aws:iam::*111122223333*:role/*role-name*** 行。
   - **--resolve-conflicts** *PRESERVE* 选项保留附加组件的现有值。如果您为附加组件设置设定了自定义值，但未使用此选项，则 Amazon EKS 会使用其默认值覆盖您的值。如果您使用此选项，那么我们建议您在更新生产集群上的附加组件之前，先测试非生产集群上所有更改的字段和值。如果您将该值改为 overwrite，则所有设置都将更改为 Amazon EKS 的默

认值。如果您为任何设置设定了自定义值，这些值可能会被 Amazon EKS 的默认值覆盖。如果您将该值改为 none，Amazon EKS 不会更改任何设置的值，但更新可能会失败。如果更新失败，您会收到一条帮助您解决冲突的错误消息。

- 如果要删除所有自定义配置，请使用 *--configuration-values '{}'* 选项执行更新。这会将所有自定义配置设置回默认值。如果不想更改自定义配置，请勿提供 *--configuration-values* 标志。如果想要调整自定义配置，请将 *{}* 替换为新参数。要查看参数列表，请参阅创建附加组件部分的查看配置架构步骤。

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

5. 检查更新的状态。将 *my-cluster* 替换为您的集群名称，将 *vpc-cni* 替换为您正在更新的附加组件的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

示例输出如下。

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
[...]
```

当状态为 ACTIVE 时更新完成。

删除附加组件

当您删除 Amazon EKS 附加组件时：

- 附加组件提供的功能不停机。
- 如果您使用的是服务账户的 IAM 角色（IRSA），并且附加组件有与其关联的 IAM 角色，则不会移除该 IAM 角色。

- 如果您使用的是容器组身份，则附加组件拥有的任何容器组身份关联都将被删除。如果将 `--preserve` 选项指定为 AWS CLI，则关联将被保留。
- Amazon EKS 停止管理附加组件的设置。
- 控制台停止通知您新版本何时可用。
- 您无法使用任何 AWS 工具或 API 更新附加组件。
- 您可以选择将附加软件留在集群上，以便您自行管理，您也可以从集群中删除附加软件。只有当集群上没有资源依赖于附加组件提供的功能时，您才能从集群中删除附加软件。

您可以使用 `eksctl`、AWS Management Console或 AWS CLI 从集群中删除 Amazon EKS 附加组件。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

使用 **`eksctl`** 删除 Amazon EKS 附加组件

1. 确定集群上目前安装的附加组件。将 *`my-cluster`* 替换为您的集群名称。

```
eksctl get addon --cluster my-cluster
```

示例输出如下。

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
[...]
```

您的输出可能会有所不同，这取决于您的集群上安装的附加组件和版本。

2. 删除附加组件。将 *`my-cluster`* 替换为您的集群名称，将 *`name-of-add-on`* 替换为上一步的输出中返回的要删除的附加组件名称。如果您删除了 ***`--preserve`*** 选项，则除了 Amazon EKS 不再管理附加组件外，附加软件也会从集群中删除。

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

AWS Management Console

使用 AWS Management Console 删除 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），然后选择要删除其 Amazon EKS 附加组件的集群的名称。
3. 选择附加组件选项卡。
4. 选中附加组件框右上角的复选框，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，但想要在集群上保留附加组件，以便您自行管理附加组件的所有设置，请选择 Preserve on the cluster（保留在集群上）。键入附加组件名称，然后选择 Remove（删除）。

AWS CLI

先决条件

您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

使用 AWS CLI 删除 Amazon EKS 附加组件

1. 查看已安装的附加组件列表。将 *`my-cluster`* 替换为您的集群名称。

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

示例输出如下。

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

2. 删除已安装的附加组件。将 *`my-cluster`* 替换为您的集群名称，将 *`name-of-add-on`* 替换为要删除的附加组件的名称。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 name-of-addon --
preserve
```

简短的示例输出如下。

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
[...]
```

3. 检查删除的状态。将 *my-cluster* 替换为您的集群名称，将 *name-of-addon* 替换为您正在删除的附加组件的名称。

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

删除附加组件后，示例输出如下。

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

检索插件版本兼容性

使用 describe-addon-verisions API 列出 EKS 插件的可用版本，以及每个插件版本支持哪些 Kubernetes 版本。

检索插件版本兼容性（AWS CLI）

1. 验证 AWS CLI 是否已安装并在使用 `aws sts get-caller-identity`。如果此命令不起作用，请了解如何开始使用 AWS CLI。
2. 确定要检索其版本兼容性信息的插件的名称，例如 `amazon-cloudwatch-observability`。
3. 确定集群的 Kubernetes 版本，例如 `1.28`。
4. 使用 AWS CLI 检索与集群的 Kubernetes 版本兼容的插件版本。

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.29
```

示例输出如下。

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "v1.5.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.28",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
[...]
```

此输出显示插件版本 `v1.5.0-eksbuild.1` 与 Kubernetes 集群版本 `1.28` 兼容。

Kubernetes 字段管理

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。

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。

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。
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。

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
[...]
```

使用容器组身份将 IAM 角色附加到 Amazon EKS 附加组件

某些 Amazon EKS 附加组件需要 IAM 角色权限才能调用 AWS API。例如，Amazon VPC CNI 附加组件会调用某些 AWS API 来配置您账户中的网络资源。需要使用 AWS IAM 对这些附加组件授予权限。更具体地说，运行附加组件的容器组（Pod）的服务账户需要与具有充分 IAM 策略的 IAM 角色相关联。

向集群工作负载授予 AWS 权限的建议方法是使用 Amazon EKS 功能容器组身份。您可以使用容器组身份关联将附加组件的服务账户映射到 IAM 角色。如果某个容器组使用具有关联的服务账户，Amazon EKS 会在容器组的容器中设置环境变量。环境变量配置 AWS SDK（包括 AWS CLI）使用 EKS 容器组身份凭证。了解有关 EKS 容器组身份的更多信息。

Amazon EKS 附加组件可以帮助管理与该附加组件对应的容器组身份关联的生命周期。例如，您可以在单个 API 调用中创建或更新 Amazon EKS 附加组件和必要的容器组身份关联。Amazon EKS 还提供用于检索建议 IAM 策略的 API。

建议的用量：

1. 确认已在您的集群上设置 Amazon EKS 容器组身份代理。
2. 使用 `describe-addon-versions` AWS CLI 操作确定要安装的附加组件是否需要 IAM 权限。如果 `requiresIamPermissions` 标志是 `true`，则应使用 `describe-addon-configurations` 操作来确定附加组件所需的权限。响应中将包含建议的托管 IAM 策略列表。
3. 使用 CLI `describe-addon-configuration` 操作检索 Kubernetes 服务账户的名称和建议的 IAM 策略。根据您的安全要求评估建议策略的范围。
4. 使用建议的权限策略和容器组身份所需的信任策略创建 IAM 角色。有关更多信息，请参阅 创建 EKS 容器组身份关联。
5. 使用该 CLI 创建或更新 Amazon EKS 附加组件。指定至少一个容器组身份关联。容器组身份关联是（1）Kubernetes 服务账户的名称，以及（2）IAM 角色的 ARN。

注意事项：

- 使用附加组件 API 创建的容器组身份关联归相应的附加组件所有。如果您删除该附加组件，则容器组身份关联也会被删除。在使用 AWS CLI 或 API 删除附加组件时，您可以通过使用 `preserve` 选项来防止这种级联删除。如有必要，您也可以直接更新或删除容器组身份关联。附加组件不能取得现有容器组身份关联的所有权。您必须删除现有关联，然后使用附加组件创建或更新操作重新创建该关联。
- Amazon EKS 建议使用容器组身份关联来管理附加组件的 IAM 权限。仍然支持之前的方法（服务账户的 IAM 角色（IRSA））。您可以为附加组件同时指定 IRSA `serviceAccountRoleArn` 和容器组身份关联。如果在集群上安装了 EKS 容器组身份代理，则 `serviceAccountRoleArn` 将被忽略，并且 EKS 将使用提供的容器组身份关联。如果未启用容器组身份，则将使用 `serviceAccountRoleArn`。
- 如果您更新现有附加组件的容器组身份关联，Amazon EKS 会启动附加组件容器组（Pod）的滚动重新启动。

检索有关附加组件的 IAM 信息

您可以使用 AWS CLI 来确定（1）某个附加组件是否需要 IAM 权限，以及（2）该附加组件的建议 IAM 策略。

检索有关 Amazon EKS 附加组件（AWS CLI）的 IAM 信息

1. 确定要安装的附加组件的名称以及集群的 Kubernetes 版本。[了解可用 Amazon EKS 附加组件的详情。]
2. 使用 AWS CLI 确定该附加组件是否需要 IAM 权限。

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
--kubernetes-version <kubernetes-version>
```

例如：

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

审查以下示例输出。请注意，`requiresIamPermissions` 的值为 `true`，即默认的附加组件版本。在检索建议的 IAM 策略时，您需要指定附加组件版本。

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
```

```
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. 如果附加组件需要 IAM 权限，请使用 AWS CLI 检索建议的 IAM 策略。

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

例如：

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

审查以下输出。记下 recommendedManagedPolicies。

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
]
```

4. 创建 IAM 角色并附加托管策略。或者，审查托管策略并根据需要缩小权限范围。审查创建用于 EKS 容器组身份的 IAM 角色的说明。

使用 IAM 角色更新附加组件

更新 Amazon EKS 附加组件以使用容器组身份关联（AWS CLI）

1. 确定：

   - `cluster-name` – 要在其上安装附加组件的 EKS 集群的名称。
   - `addon-name` – 要安装的 Amazon EKS 附加组件的名称。
   - `service-account-name` – 该附加组件使用的 Kubernetes 服务账户的名称。
   - `iam-role-arn` – 对附加组件拥有足够权限的 IAM 角色的 ARN。IAM 角色必须具有 EKS 容器组身份所需的信任策略。

2. 使用 AWS CLI 更新附加组件。您还可以在创建附加组件时使用相同的 `--pod-identity-assocations` 语法指定容器组身份关联。请注意，如果您在更新附加组件时指定容器组身份关联，则所有之前的容器组身份关联都将被覆盖。

   ```
   aws eks update-addon --cluster-name <cluster-name> \
   --addon-name <addon-name> \
   --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
   ```

   例如：

   ```
   aws eks update-addon --cluster-name mycluster \
   --addon-name aws-ebs-csi-driver \
   --pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
   ```

3. 验证是否已创建容器组身份关联：

   ```
   aws eks list-pod-identity-associations --cluster-name <cluster-name>
   ```

   如果成功，您应看到与以下内容类似的输出。记下 EKS 附加组件的 OwnerARN。

   ```
   {
       "associations": [
           {
               "clusterName": "mycluster",
               "namespace": "kube-system",
               "serviceAccount": "ebs-csi-controller-sa",
   ```

```
            "associationArn": "arn:aws:eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws:eks:us-west-2:123456789012:addon/mycluster/aws-
ebs-csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

从附加组件中移除关联

从 Amazon EKS 附加组件（AWS CLI）中移除所有容器组身份关联

1. 确定：
   - `cluster-name` – 要在其上安装附加组件的 EKS 集群的名称。
   - `addon-name` – 要安装的 Amazon EKS 附加组件的名称。
2. 更新附加组件以指定容器组身份关联的空数组。

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

对 EKS 附加组件的容器组身份进行故障排除

如果您的附加组件在尝试 AWS API、SDK 或 CLI 操作时遇到错误，请确认以下内容：

- 容器组身份代理已安装在您的集群中。
  - 审查如何设置容器组身份代理。
- 该附加组件具有有效的容器组身份关联。
  - 使用 AWS CLI 检索附加组件使用的服务账户名称的关联。

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- 预期的 IAM 角色具有 EKS 容器组身份所需的信任策略。
  - 使用 AWS CLI 检索附加组件的信任策略。

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- 预期的 IAM 角色具有该附加组件的必要权限。
  - 使用 AWS CloudTrail 审查 `AccessDenied` 或 `UnauthorizedOperation` 事件。
- 容器组身份关联中的服务账户名称与附加组件使用的服务账户名称相匹配。
  - 审查附加组件的文档以确定服务账户名称。

在部署期间验证容器镜像

如果您在部署时使用 AWS Signer 并且想要验证已签名的容器镜像，则可以使用以下解决方案之一：

- Gatekeeper 和 Ratify – 将 Gatekeeeper 用作准入控制器，并将配置了 AWS Signer 插件的 Ratify 作为用于验证签名的 Webhook。
- Kyverno – 配置有用于验证签名的 AWS Signer 插件的 Kubernetes 策略引擎。

> **Note**
>
> 在验证容器镜像签名之前，请根据所选准入控制器的要求配置 Notation 信任存储和信任策略。

使用 Elastic Fabric Adapter 的 Machine Learning 培训

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AWS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AWS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 400Gbps 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 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 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` 创建您的节点组，`eksctl` 还可以为您创建集群。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。
- 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 Amazon VPC CNI plugin for Kubernetes 版本的更多信息，请参阅 使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件。

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 AWS 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 AWS 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 AWS 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 AWS 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
```

示例输出如下。

```
us-west-2a    us-west-2c    us-west-2b
```

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 AWS CLI 和 AWS CloudFormation 创建节点组。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。

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.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p4d.24xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    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。

AWS CLI and AWS 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《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 AWS 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=us-west-2a" \
    --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 中没有空间再创建其他子网，则可以向该 VPC 中添加 CIDR 块并在新 CIDR 块中创建子网，或者在新 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
```

示例输出如下。

```
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 AWS CloudFormation 模板。

```
curl -O 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 AWS CloudFormation 堆栈所创建的资源创建您的节点组。

a. 从已部署的 AWS 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. 如果您选择了 GPU 实例类型，则必须部署 适用于 Kubernetes 的 NVIDIA 设备插件。将 *vX.X.X* 替换为您需要的 NVIDIA/k8s-device-plugin 版本，然后运行以下命令。

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-
plugin/vX.X.X/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。

```
helm repo add eks https://aws.github.io/eks-chart
helm install aws-efa-k8s-device-plugin --namespace kube-system eks/aws-efa-k8s-
device-plugin
```

（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/
v2beta1/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 NVIDIA 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 AWS 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
```

   示例输出如下。

```
已创建 mpijob.kubeflow.org/nccl-tests-efa
```

2. 查看您运行的 Pods。

```
kubectl get pods
```

   示例输出如下。

```
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 AWS 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 AWS Inferentia 芯片提供支持，这些芯片由AWS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 AWS Neuron 部署到容器，而 AWS 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。AWSNeuron 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 `eksctl` 文档中的 Installation。
- 在计算机上安装 `kubectl`。有关更多信息，请参阅 安装或更新 `kubectl`。
- （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 AWS 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 AWS Deep Learning AMI 开发人员指南中的带有 DLAMI 的 AWS 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 AWS Deep Learning Containers 提供的 TensorFlow。容器内部是 AWS 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 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。

   ```
   aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --
   region us-west-2
   ```

   ```
   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
```

示例输出如下。

```
[[(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 AWS Management Console 查看有关 Kubernetes 资源的信息。

- Kubernetes 控制面板是适用于 Kubernetes 集群的基于 Web 的通用用户界面。用户能通过它管理集群中运行的应用程序并对其进行故障排除，以及管理集群本身。有关更多信息，请参阅 GitHub 存储库中的 Kubernetes 控制面板。
- 安装 Kubernetes Metrics Server – 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）使用。在本主题中，您将了解如何安装 Metrics Server。
- 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。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 – 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 – 您的AWS账户对于每项AWS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。

成本监控

成本监控是管理 Amazon EKS 上的 Kubernetes 集群的重要方面。通过了解集群成本，您可以优化资源利用率，设定预算，并对部署做出数据驱动型决策。Amazon EKS 提供两种成本监控解决方案，每种解决方案都有其独特的优势，可帮助您有效跟踪和分配成本：

Amazon EKS 的 AWS 账单拆分成本分配数据 – 此原生功能与 AWS 账单控制台无缝集成，使您能够使用与其他 AWS 服务相同的熟悉界面和工作流来分析和分配成本。通过拆分成本分配，您可以直接了解自己的 Kubernetes 成本以及其他 AWS 支出，从而更轻松地在整个 AWS 环境中全面优化成本。您还可以利用 Cost Categories 和成本异常检测等现有 AWS 账单功能，进一步增强您的成本管理能力。有关更多信息，请参阅《AWS 账单用户指南》中的了解拆分成本分配数据。

Kubecost – Amazon EKS 支持 Kubecost，后者是一种 Kubernetes 成本监控工具。Kubecost 提供一种功能丰富的 Kubernetes 原生成本监控方法，提供按 Kubernetes 资源划分的精细成本明细、成本优化建议以及开箱即用的控制面板和报告。Kubecost 还通过与 AWS 成本和使用情况报告集成来检索准确的定价数据，从而确保您精确地了解自己的 Amazon EKS 成本。了解如何安装 Kubecost。

AWS 账单 – 拆分成本分配

使用 Amazon EKS 的 AWS 拆分成本分配数据进行成本监控

您可以使用 Amazon EKS 的 AWS 拆分成本分配数据来获得对您的 Amazon EKS 集群的精细的了解。这使您能够分析、优化并退还 Kubernetes 应用程序的成本和使用量。您可以根据 Kubernetes 应用程序消耗的 Amazon EC2 CPU 和内存资源将应用程序成本分配给各个业务部门和团队。Amazon EKS 的拆分成本分配数据可让您了解每个容器组（pod）的成本，并使您能够使用命名空间、集群和其他 Kubernetes 基元汇总每个容器组（pod）的成本数据。以下是可用于分析 Amazon EKS 成本分配数据的 Kubernetes 基元示例。

- 集群名称
- 部署
- 命名空间
- 节点
- 工作负载名称
- 工作负载类型

有关使用拆分成本分配数据的更多信息，请参阅《AWS 账单用户指南》中的了解拆分成本分配数据。

设置成本和使用情况报告

您可以在成本管理控制台、AWS Command Line Interface 或 AWS SDK 中启用 ECS 的拆分成本分配数据。

将以下内容用于拆分成本分配数据：

1. 选择拆分成本分配数据。有关更多信息，请参阅《AWS 成本和使用情况报告 用户指南》中的启用拆分成本分配数据。
2. 将数据纳入新报告或现有报告。
3. 查看报告。您可以使用账单和成本管理控制台或在 Amazon Simple Storage Service 中查看报告文件。

Kubecost

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 AWS 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 AWS 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 AWS 支持协议获取支持。

先决条件

- 现有 Amazon EKS 集群。要部署一个角色，请参阅 开始使用 Amazon EKS。集群必须具有 Amazon EC2 节点，因为您无法在 Fargate 节点上运行 Kubecost。
- 您的设备或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。
- 您的设备或 AWS 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”。
- 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”。

要安装 Kubecost

1. 确定要安装的 Kubecost 版本。您可以在 Amazon ECR Public Gallery 中的 kubecost/cost-analyzer 上查看可用的版本。有关 Kubecost 版本和 Amazon EKS 兼容性的更多信息，请参阅 Kubecost 文档中的环境要求。
2. 使用以下命令安装 Kubecost。将 *`kubecost-version`* 替换为从 ECR 中检索到的值，例如 *`1.108.1`*。

   ```
   helm upgrade -i kubecost oci://public.ecr.aws/kubecost/cost-analyzer --
   version kubecost-version \
       --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。默认情况下，安装包括本地 Prometheus 服务器和 `kube-state-metrics`。您可以按照与 Amazon EKS 成本监控集成中的文档来自定义部署，以使用 Amazon Managed Service for Prometheus。有关您可以配置的所有其他设置的列表，请参阅 GitHub 上的示例配置文件。

3. 确保所需的 Pods 正在运行。

```
kubectl get pods -n kubecost
```

示例输出如下。

```
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
```

4. 在您的设备上，启用端口转发以公开 Kubecost 控制面板。

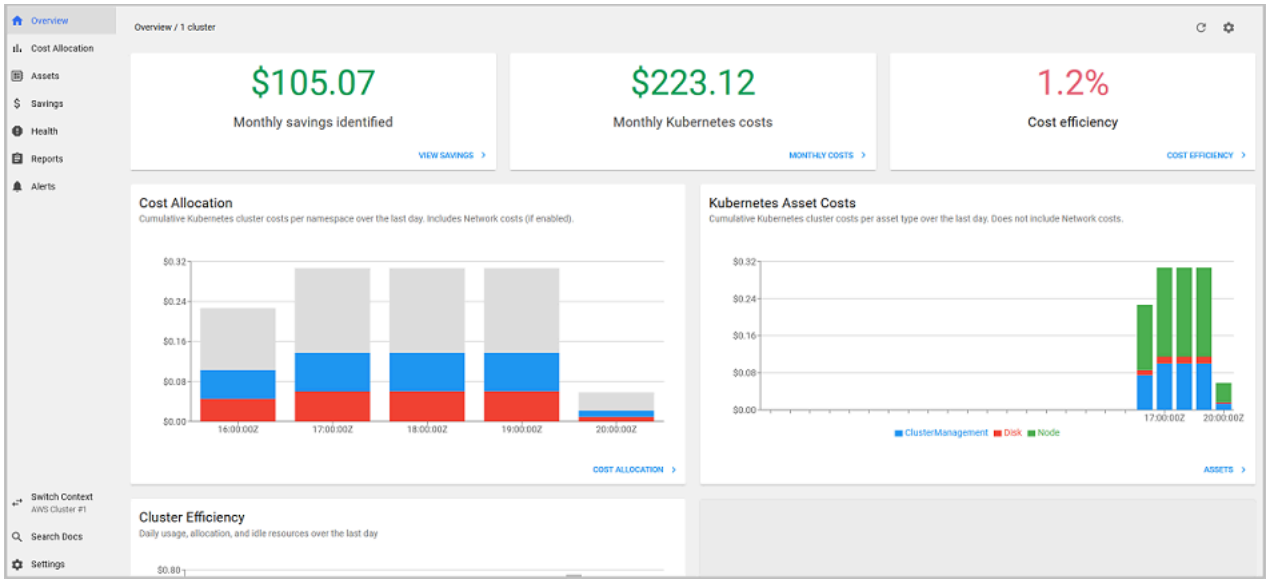
```
kubectl port-forward --namespace kubecost deployment/kubecost-cost-analyzer 9090
```

或者，您可以使用 AWS Load Balancer Controller 公开 Kubecost 并将 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。

5. 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。



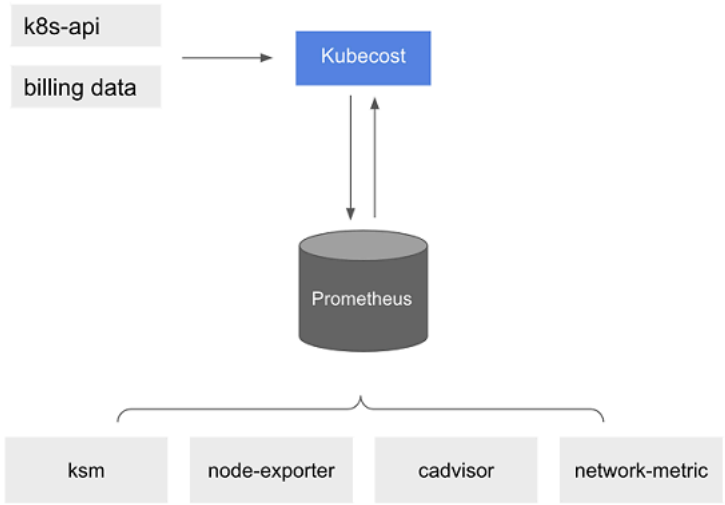
6. 要跟踪集群级别的成本，请标记您的 Amazon EKS 资源以进行计费。有关更多信息，请参阅 标记资源以便于计费。

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 AWS 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，或者参考 Kubecost Github 存储库 中的示例 Grafana JSON 模型。

- AWS 成本和使用情况报告 集成 – 要为您的 Amazon EKS 集群执行成本分配计算，Kubecost 会从 AWS 价目表 API 中检索 AWS 服务 和 AWS 资源的公开定价信息。您还可以将 Kubecost 和 AWS 成本和使用情况报告 集成，以提高特定于 AWS 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 AWS 成本和使用情况报告 集成工作原理的更多信息，请参阅 Kubecost 文档中的 AWS 云账单集成。

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
```

常见问题

请参阅有关将 Kubecost 与 Amazon EKS 配合使用的以下常见问题和答案。

Kubecost 自定义捆绑包与 Kubecost 免费版（也称为 OpenCost）之间有何区别？

AWS 与 Kubecost 合作提供了一个自定义版本的 Kubecost。此版本包括一部分商业功能，无需额外付费。有关 Kubecost 自定义捆绑包中包含的功能，请参阅下表。

功能	Kubecost 免费套餐	Amazon EKS 优化型 Kubecost 自定义捆绑包	Kubecost Enterprise
部署	用户托管	用户托管	用户托管或 Kubecost 托管（SaaS）
支持的集群数量	无限制	无限制	无限制
支持的数据库	本地 Prometheus	本地 Prometheus 或 Amazon Managed Service for Prometheus	Prometheus、Amazon Managed Service for Prometheus、Cortex 或 Thanos
数据库保留支持	15 天	无限历史数据	无限历史数据
Kubecost API 保留（ETL）	15 天	15 天	无限历史数据
集群成本可见性	单个集群	统一多集群	统一多集群

功能	Kubecost 免费套餐	Amazon EKS 优化型 Kubecost 自定义捆绑包	Kubecost Enterprise
混合云可见性	-	Amazon EKS 和 Amazon EKS Anywhere 集群	多云和混合云支持
提醒和定期报告	-	支持效率提醒、预算提醒、支出变化提醒等	支持效率提醒、预算提醒、支出变化提醒等
保存的报告	-	基于 15 天数据的报告	基于无限历史数据的报告
云账单集成	每个单独的集群都需要	AWS 自定义定价支持（包括多个集群和多个账户）	AWS 自定义定价支持（包括多个集群和多个账户）
节省建议	单集群洞察	单集群洞察	多集群洞察
治理：审计	-	-	审计历史成本事件
单点登录（SSO）支持	-	支持 Amazon Cognito	Okta、Auth 0、PingID、KeyCloak
使用 SAML **2.0** 的基于角色的访问控制（RBAC）	-	-	Okta, Auth0, PingID, Keycloak
企业培训和引导	-	-	全方位服务培训和 FinOps 引导

什么是 Kubecost API 保留（ETL）功能？

Kubecost ETL 功能可汇总和组织各种指标，以显示不同粒度级别的成本可见性（例如 `namespace-level`、`pod-level` 和 `deployment-level`）。对于自定义 Kubecost 捆绑包，客户可以获得 15 天的指标数据和洞察。

什么是提醒和定期报告功能？它包括哪些提醒和报告？

借助 Kubecost 提醒，团队可以实时获得有关 Kubernetes 支出和云支出的动态。借助定期报告，团队能够获得有关历史 Kubernetes 和云支出的自定义视图。这两者都可以使用 Kubecost UI 或 Helm 值来配置。它们支持电子邮件、Slack 和 Microsoft Teams。

保存的报告包含哪些内容？

Kubecost 保存的报告是有关成本和效率指标的预定义视图。其中包括按集群、命名空间、标签等划分的成本。

什么是云账单集成？

通过与 AWS 账单 API 集成，可让 Kubecost 显示集群之外的成本（例如 Amazon S3 成本）。此外，通过这种集成，还可让 Kubecost 根据实际账单核对 Kubecost 的集群内预测，从而可以考虑竞价型实例使用情况、实惠配套和企业折扣。

节省建议包含哪些内容？

Kubecost 提供了相关的洞察和自动化功能，可帮助用户优化其 Kubernetes 基础设施和支出。

此功能是否收费？

不收费。您可以免费使用此版本的 Kubecost。如果您需要未包括在此捆绑包中的其他 Kubecost 功能，可以通过 AWS Marketplace 或直接从 Kubecost 购买 Kubecost 的企业许可证。

是否提供支持？

是。您可以通过 联系 AWS，联系 AWS Support 团队创建支持工单。

我是否需要许可证才能使用 Amazon EKS 集成提供的 Kubecost 功能？

否。

我是否可以将 Kubecost 与 AWS 成本和使用情况报告 集成，以获得更准确的报告？

是。您可以将 Kubecost 配置为从 AWS 成本和使用情况报告 摄取数据，以获取准确的成本可见性，包括折扣、竞价型定价、预留实例定价，以及其他内容。有关更多信息，请参阅 Kubecost 文档中的 AWS 云账单集成。

此版本是否支持 Amazon EC2 上自行管理的 Kubernetes 集群的成本管理？

不支持。此版本仅与 Amazon EKS 集群兼容。

Kubecost 是否能够追踪 AWS Fargate 上的 Amazon EKS 的成本?

Kubecost 已尽最大努力显示 Fargate 上的 Amazon EKS 的集群成本可见性，但准确性低于 Amazon EC2 上的 Amazon EKS。这主要是由于您的使用计费方式不同。使用 Fargate 上的 Amazon EKS，将针对消耗的资源为您计费。使用 Amazon EC2 节点上的 Amazon EKS，将针对预调配的资源为您计费。Kubecost 将根据节点规格（包括 CPU、RAM 和临时存储）计算 Amazon EC2 节点的成本。使用 Fargate，将根据为 Fargate Pods 请求的资源计算成本。

我如何获得 Kubecost 的更新和新版本？

您可以使用标准 Helm 升级程序升级您的 Kubecost 版本。最新版本位于 Amazon ECR Public Gallery 中。

是否支持 **`kubectl-cost`** CLI？ 我如何安装它？

支持。`kubectl-cost` 是一个通过 Kubecost（Apache 2.0 许可证）使用的开源工具，提供针对 Kubernetes 成本分配指标的 CLI 访问权限。要安装 `kubectl-cost`，请参阅 GitHub 上的安装。

是否支持 Kubecost 用户界面？ 我如何访问它？

Kubecost 提供了一个 Web 控制面板，您可以通过 `kubectl` 端口转发、入口或负载均衡器访问该控制面板。您也可以使用 AWS Load Balancer Controller 公开 Kubecost，并使用 Amazon Cognito 进行身份验证、授权和用户管理。有关更多信息，请参阅 AWS 博客上的如何使用应用程序负载均衡器和 Amazon Cognito 对您的 Kubernetes Web 应用程序的用户进行身份验证。

是否支持 Amazon EKS Anywhere？

否。

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 或 Kubernetes 控制面板。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**
>
> 这些指标是用于时间点分析，不是历史分析的准确来源。它们不能用作监控解决方案或用于其他非自动扩缩目的。有关监控工具的信息，请参阅 Amazon EKS 中的可观察性。

部署 Metrics Server

1. 使用以下命令部署 Metrics Server：

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

如果您使用的是 Fargate，则需要更改此文件。在默认配置中，指标服务器使用端口 10250。此端口在 Fargate 上保留。将 components.yaml 中对端口 10250 的引用替换为其他端口，例如 10251。

2. 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
```

示例输出如下。

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

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 集群创建或更新 `kubeconfig` 文件后再继续。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 | cut -d + -f 1
```

示例输出如下。

```
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

> **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AWS资源分配的标记。每个标签都包含一个键和一个可选值。

借助标签，您可以对 AWS 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，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 AWS Identity and Access Management（IAM），则可以控制您的AWS账户中的哪些用户拥有管理标签的权限。

标记资源

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。
- 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 AWS CLI、Amazon EKS API 或 AWS 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 AWS 服务 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 AWS 使用。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 AWS Billing 用户指南中的 AWS 成本和使用情况报告。

通过 AWS 生成的成本分配标签（特别是 `aws:eks:cluster-name`），您可以在 Cost Explorer 中按单个 Amazon EKS 集群细分 Amazon EC2 实例成本。但此标签不会捕获控制面板开支。该标签会自动添加到参与 Amazon EKS 集群的 Amazon EC2 实例中。无论实例是使用 Amazon EKS 托管节点组、Karpenter 还是直接通过 Amazon EC2 预调配的，都会发生此行为。此特定标签不会计入 50 个标签的限制。要使用该标签，账户所有者必须在 AWS Billing 控制台中或者通过使用 API 来激活它。当 AWS Organizations 管理账户所有者激活该标签时，还将同时为所有组织成员账户激活该标签。

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 AWS 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 AWS 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 AWS 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 AWS 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《AWS 标签编辑器用户指南》中的使用标签编辑器为 AWS 资源添加标签。

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。

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 AWS 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 AWS 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	AWS CLI	AWS Tools for Windows PowerShell	API 操作
添加或覆盖一个或多个标签。	`tag-resource`	`Add-EKSResourceTag`	`TagResource`

任务	AWS CLI	AWS Tools for Windows PowerShell	API 操作
删除一个或多个标签。	untag-resource	Remove-EKSResourceTag	UntagResource

以下示例说明如何使用AWS CLI给资源加标签或取消标签。

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	AWS CLI	AWS Tools for Windows PowerShell	API 操作	**eksctl**
创建集群	create-cluster	New-EKSCluster	CreateCluster	create cluster

任务	AWS CLI	AWS Tools for Windows PowerShell	API 操作	**eksctl**
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.html	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。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《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 AWS 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《服务限额用户指南》中的什么是服务限额？。借助服务限额集成，您可以使用 AWS Management Console 和 AWS CLI 快速查找 Amazon EKS 和 AWS Fargate 服务限额的值。

AWS Management Console

使用 AWS Management Console 查看 Amazon EKS 和 Fargate 服务配额

1. 访问 https://console.aws.amazon.com/servicequotas/，打开服务限额控制台。
2. 在左侧导航窗格中，选择 AWS 服务。
3. 从 AWS 服务 列表中，搜索并选择 Amazon Elastic Kubernetes Service (Amazon EKS) 或 AWS Fargate。

   在 Service Quotas（服务限额）列表中，您可以查看服务限额名称、应用的值（如果该值可用）、AWS 默认限额以及限额值是否可调整。
4. 要查看有关服务限额的其他信息（如描述），请选择限额名称。
5. （可选）要请求增加配额，请选择要增加的配额，选择 Request quota increase（请求增加配额），输入或选择所需信息，然后选择 Request（请求）。

要使用 AWS 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。

AWS CLI

使用 AWS 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 AWS 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 AWS CLI 进一步处理服务限额，请参阅《AWS CLI 命令参考》中的 `service-quotas`。要请求提高配额，请参阅 AWS 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
每个集群的访问条目	每个受支持的区域：3000 个	不支持	每个集群的最大访问条目数。
集群	每个受支持的区域：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	每个受支持的区域：4 个	不支持	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）。
EKS Anywhere Enterprise 订阅	每个受支持的区域：10 个	是	当前区域的此账户中 EKS Anywhere Enterprise 订阅的最大数量。
每个集群的 Fargate 配置文件	每个受支持的区域：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	每个受支持的区域：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	每个受支持的区域：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	每个受支持的区域：450 个	是	每个托管节点组的最大节点数。
每个集群的公有端点访问 CIDR 范围数	每个受支持的区域：40 个	不支持	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	每个受支持的区域：10 个	是	当前区域中的此账户中的注册集群的最大数量。

名称	默认值	可调整	描述
每个 Fargate 配置文件的选择器数	每个受支持的区域：5 个	是	每个 Fargate 配置文件的最大选择器数量。

> **Note**
>
> 默认值是由 AWS 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。

这些服务限额列于服务限额控制台中 Amazon Elastic Kubernetes Service (Amazon EKS) 下。对于显示为可调整的值，要请求提高限额，请参阅服务限额用户指南中的请求提高限额。

AWS Fargate 服务限额

服务限额控制台中的 AWS Fargate 服务列出了多个服务限额。下表仅描述了适用于 Amazon EKS 的限额。您可以配置警报，以在用量接近服务限额时向您发出警报。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。

新 AWS 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 AWS 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，请参阅服务限额用户指南中的请求增加限额。

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。

> **Note**
>
> 默认值是由 AWS 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 Pods 启动率限额。有关更多信息，请参阅《Amazon ECS 指南》中的 AWS Fargate 节流配额。

Amazon EKS 中的安全性

AWS 十分重视云安全性。作为 AWS 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 AWS 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 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。对于 Amazon EKS，AWS 负责 Kubernetes 控制面板，其中包括控制面板节点和 etcd 数据库。作为 AWS 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AWS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AWS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

- Amazon Elastic Kubernetes Service 的合规性验证
- Amazon EKS 中的恢复能力
- Amazon EKS 中的基础设施安全性
- Amazon EKS 中的配置和漏洞分析
- Amazon EKS 的安全最佳实践
- 容器组（pod）安全策略
- 容器组（pod）安全策略 (PSP) 移除常见问题
- 将 Kubernetes 与 AWS Secrets Manager 密钥结合使用
- Amazon EKS Connector 注意事项

证书签名

Kubernetes 证书 API 会自动化 X.509 凭证预置。API 具有一个命令行界面，供 Kubernetes API 客户端从证书颁发机构（CA）请求和获取 X.509 证书。您可以使用 `CertificateSigningRequest`（CSR）资源请求指示签署人对证书进行签名。您的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 `legacy-unknown` 值作为 `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。如果想使用 Amazon EKS CA 在集群上生成证书，则您必须使用自定义签署人。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。使用现有 `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：

- 信任分配：无。在 Kubernetes 集群中，此签署人没有标准的信任或分配。
- 允许的主题：任何
- 允许的 x509 扩展：遵循 subjectAltName 和密钥使用扩展，并丢弃其他扩展
- 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

> **Note**
>
> 不支持客户端证书签名。

- 到期/证书使用寿命：1 年（默认值和最大值）
- 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. 为 CSR 请求生成 base64 值，并将其存储在变量中，以便在后续步骤中使用。

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "\n")
```

4. 要创建名为 `mycsr.yaml` 的文件，请运行以下命令。在以下示例中，`beta.eks.amazonaws.com/app-serving` 是 `signerName`。

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
```

```
    - server auth
EOF
```

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
```

示例输出如下。

```
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
```

将集群升级到 Kubernetes 1.24 之前的证书签名注意事项

在 Kubernetes 1.23 及早期版本中，具有不可验证的 IP 和 DNS 使用者备用名称（SAN）的 kubelet 服务证书会自动使用不可验证的 SAN 进行颁发。预置的证书中省略了 SAN。在 1.24 及更高版本的集群中，如果无法验证 SAN，则不会颁发 kubelet 服务证书。这会阻止 kubectl exec 和 kubectl logs 命令发挥作用。

将集群升级到 1.24 之前，请完成以下步骤，以确定集群中是否存在尚未批准的证书签名请求（CSR）：

1. 运行以下命令。

```
kubectl get csr -A
```

示例输出如下。

```
NAME        AGE   SIGNERNAME                        REQUESTOR
                          REQUESTEDDURATION   CONDITION
csr-7znmf   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-42-149.region.compute.internal      <none>
 Approved
csr-9xx5q   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-65-38.region.compute.internal      <none>
 Approved, Issued
```

如果返回的输出显示 CSR，其具有为节点 Approved 而非 Issued 的 kubernetes.io/kubelet-serving 签署人，则您需要批准该请求。

2. 手动批准 CSR。将 csr-7znmf 替换为您自己的值。

```
kubectl certificate approve csr-7znmf
```

要在将来自动批准 CSR，我们建议您编写一个批准控制器，该控制器可自动验证和批准包含 Amazon EKS 无法验证的 IP 或 DNS SAN 的 CSR。

适用于 Amazon EKS 的身份和访问管理

AWS Identity and Access Management（IAM）是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 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 AWS 服务。

受众

如何使用 AWS 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有助于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 IAM 故障排除。

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。

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。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center（IAM Identity Center）用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的 如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 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《AWS IAM Identity Center 用户指南》中的多重身份验证 和《IAM 用户指南》中的 在 AWS 中使用多重身份验证（MFA）。

AWS 账户 根用户

当您创建 AWS 账户 时，最初使用的是一个对账户中所有 AWS 服务 和资源拥有完全访问权限的登录身份。此身份称为 AWS 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。有关需要您以根用户身份登录的任务的完整列表，请参阅 IAM 用户指南 中的 需要根用户凭证的任务。

IAM 用户和群组

IAM 用户是 AWS 账户 内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有

一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

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 AWS 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 AWS Management Console 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以担任角色。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：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 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 AWS 服务，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 AWS 服务 使用其它 AWS 服务中的特征。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 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。
  - 转发访问会话：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。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启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

  - 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分派的 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 AWS 服务 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 AWS 服务 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 AWS 账户 中，并由该服务拥有。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 AWS CLI 或 AWS 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 AWS 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 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 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，AWS 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 AWS 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 AWS 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。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。然后，管理员可以向角色添加 IAM 策略，并且用户可以代入角色。

IAM 策略定义操作的权限，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 AWS Management Console、AWS CLI 或 AWS 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 策略。

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 AWS 账户 中的多个用户、组和角色的独立策略。托管策略包括 AWS 托管策略和客户管理型策略。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、联合用户或 AWS 服务。

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 AWS 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）、AWS 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

AWS 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特征，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 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有特征，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（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 AWS 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 AWS 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测逻辑。

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AWS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AWS服务。

主题

- Amazon EKS 基于身份的策略
- Amazon EKS 基于资源的策略
- 基于 Amazon EKS 标签的授权
- Amazon EKS IAM 角色

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 AWS 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 AWS 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关联操作的权限。

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 AWS 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 资源名称（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 AWS 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 AWS 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 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 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 策略。

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 AWS 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 定义的条件。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。

创建 Amazon EKS 集群时，将为创建集群的 IAM 主体自动授予 Amazon EKS 控制面板中基于集群角色的访问控制（RBAC）配置中的 `system:masters` 权限。该主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主体。要授予其他 IAM 主体与集群进行交互的能力，请编辑 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 授予对 Kubernetes API 的访问权限 。

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。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 AWS 账户中具有特定权限的实体。

将临时凭证用于 Amazon EKS

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。可以通过调用 AWS 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 AWS 服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在 IAM 账户中，并归该服务所有。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。

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他服务中的资源以代表您完成操作。服务角色显示在 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 和 Amazon EKS 节点 IAM 角色。

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AWS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 和检查现有节点角色。

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 AWS Management Console、AWS CLI 或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，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，将为创建集群的 IAM 主体自动授予 Amazon EKS 控制面板中基于集群角色的访问控制（RBAC）配置中的 `system:masters` 权限。该主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主体。要授予其他 IAM 主体与集群进行交互的能力，请编辑 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 授予对 Kubernetes API 的访问权限 。

主题

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 AWS 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：

- AWS 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 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 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 AWS 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 AWS 托管策略或工作职能的 AWS 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（AWS 服务例如 AWS 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 策略，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 策略语言 (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。
- Require multi-factor authentication（MFA）[需要多重身份验证（MFA）] – 如果您所处的场景要求您的 AWS 账户 中有 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 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

使用 Amazon EKS 控制台

要访问 Amazon EKS 控制台，IAM 主体必须具有一组最低的权限。这些权限允许主体列出和查看有关您的 AWS 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主体，控制台将无法按预期正常运行。

为确保 IAM 主体仍可以使用 Amazon EKS 控制台，请用唯一名称创建一个策略，如 `AmazonEKSAdminPolicy`。将策略附加到主体。有关更多信息，请参阅《IAM 用户指南》中的添加和移除 IAM 身份权限。

> **Important**
>
> 以下示例策略将允许主体查看控制台中配置选项卡上的信息。要在 AWS Management Console 中的概述和资源选项卡上查看信息，该主体还需要 Kubernetes 权限。有关更多信息，请参阅 所需的权限。

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

对于仅调用 AWS CLI 或 AWS API 的主体，您不需要允许最低控制台权限。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 IAM 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 AWS CLI 或 AWS 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": "*"
        }
    ]
}
```

在 AWS Cloud 上创建 Kubernetes 集群

此示例策略包含在 *us-west-2* AWS 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 AWS 区域 替换为要在其中创建集群的 AWS 区域。如果 AWS Management Console 中显示警告您的策略中的操作不支持资源级权限，因而要求您选择 **All resources**，您可以放心地忽略该警告。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:us-west-2:111122223333: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"
        }
    ]
}
```

在 Outpost 上创建本地 Kubernetes 集群

此示例策略包括在 *us-west-2* AWS 区域 中的 Outpost 上创建名为 *my-cluster* 的 Amazon EKS 本地集群所需的最低权限。您可以将 AWS 区域 替换为要在其中创建集群的 AWS 区域。如果 AWS Management Console 中显示警告您的策略中的操作不支持资源级权限，因而要求您选择 **`All resources`**，您可以放心地忽略该警告。如果您的账户已经有 `AWSServiceRoleForAmazonEKSLocalOutpost` 角色，您可以从策略中删除 `iam:CreateServiceLinkedRole` 操作。如果您曾经在您的账户中的 Outpost 上创建过 Amazon EKS 本地集群，则此角色已存在，除非您将其删除。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
```

```
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ]
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        },
    ]
}
```

更新 Kubernetes 集群

此示例策略包含在 us-west-2 AWS 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:us-west-2:111122223333: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 主体必须能够列出和描述集群，才能使用 `update-kubeconfig` AWS 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 AWS 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AWS服务所需的所有权限。

主题

- 使用 Amazon EKS 集群的角色
- 使用 Amazon EKS 节点组的角色
- 使用 Amazon EKS Fargate 配置文件的角色
- 使用角色将 Kubernetes 集群连接到 Amazon EKS
- 使用 Outpost 上的 Amazon EKS 本地集群的角色

使用 Amazon EKS 集群的角色

Amazon Elastic Kubernetes Service 使用 AWS 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AWS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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您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 AWS 服务，并查找 Service-linked role（服务相关角色）列中为 Yes（是）的服务。选择是，可转到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。

`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`

您必须配置权限，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

您无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS API 中创建集群时，Amazon EKS 会为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您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，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。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 和 删除 Fargate 配置文件。
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、AWS CLI 或 AWS 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 端点和配额。

使用 Amazon EKS 节点组的角色

Amazon EKS 使用 AWS 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AWS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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您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 AWS 服务，并查找 Service-linked role（服务相关角色）列中为 Yes（是）的服务。选择是，可转到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`

您必须配置权限，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

您无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS 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 (AWS API) 中创建服务相关角色

您无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS API 中创建托管节点组时，Amazon EKS 会为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您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，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。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、AWS CLI 或 AWS 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 端点和配额。

使用 Amazon EKS Fargate 配置文件的角色

Amazon EKS 使用 AWS 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AWS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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您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 AWS 服务，并查找 Service-linked role（服务相关角色）列中为 Yes（是）的服务。选择是，可转到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 Pods 所需的 VPC 联网。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

您必须配置权限，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

您无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS 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 (AWS API) 中创建服务相关角色

您无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS API 中创建 Fargate 配置文件时，Amazon EKS 将为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您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，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。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、AWS CLI 或 AWS 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 端点和配额。

使用角色将 Kubernetes 集群连接到 Amazon EKS

Amazon EKS 使用 AWS 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AWS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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您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 AWS 服务，并查找 Service-linked role（服务相关角色）列中为 Yes（是）的服务。选择是，可转到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`

您必须配置权限，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 AWS Management Console、AWS CLI、`eksctl` 或 AWS API 中连接集群时，Amazon EKS 会为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您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，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。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、AWS CLI 或 AWS 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。

使用 Outpost 上的 Amazon EKS 本地集群的角色

Amazon Elastic Kubernetes Service 使用 AWS 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AWS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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您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 AWS 服务，并查找 Service-linked role（服务相关角色）列中为 Yes（是）的服务。选择是，可转到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 AWSServiceRoleForAmazonEKSLocalOutpost 的服务相关角色 – 该角色允许 Amazon EKS 管理您的账户中的本地集群。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 Amazon EC2 实例。

> **Note**
>
> AWSServiceRoleForAmazonEKSLocalOutpost 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。

AWSServiceRoleForAmazonEKSLocalOutpost 服务相关角色信任以下服务代入该角色：

- outposts.eks-local.amazonaws.com

角色权限策略允许 Amazon EKS 对指定资源完成以下操作：

- AmazonEKSServiceRolePolicy

您必须配置权限，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

您无需手动创建服务相关角色。当您在 AWS Management Console、AWS CLI 或 AWS API 中创建集群时，Amazon EKS 会为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您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 AWSServiceRoleForAmazonEKSLocalOutpost 服务相关角色。在创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以使用 IAM 编辑角色的说明。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 **`AWSServiceRoleForAmazonEKSLocalOutpost`** 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. 请在左侧导航窗格中，选择 Amazon EKS Clusters（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 和 删除 Fargate 配置文件。
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、AWS CLI 或 AWS API 删除 `AWSServiceRoleForAmazonEKSLocalOutpost` 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 端点和配额。

Amazon EKS 集群 IAM 角色

对于每个集群，均需要 Amazon EKS 集群 IAM 角色。由 Amazon EKS 管理的 Kubernetes 集群使用此角色来管理节点，而传统 Cloud Provider 使用此角色为服务创建带有 Elastic Load Balancing 的负载均衡器。

必须先使用以下任一 IAM policy 创建 IAM 角色，然后才能创建 Amazon EKS 集群：

- AmazonEKSClusterPolicy
- 自定义 IAM policy。随后的最低权限允许 Kubernetes 集群管理节点，但不允许传统 Cloud Provider 创建带有 Elastic Load Balancing 的负载均衡器。您的自定义 IAM policy必须至少具有以下权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 在 2023 年 10 月 3 日之前，每个集群的 IAM 角色都必须有 AmazonEKSClusterPolicy。在 2020 年 4 月 16 日之前，AmazonEKSServicePolicy 和 AmazonEKSClusterPolicy 是必需的，建议的角色名称是 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 AmazonEKSServicePolicy 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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 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. 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 AWS Management Console 或 AWS CLI。

AWS Management Console

在 IAM 控制台中创建 Amazon EKS 集群角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择 Roles，然后选择 Create role。
3. 在受信任的实体类型下，选择 AWS 服务。
4. 从其他 AWS 服务 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)。

AWS 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 AWS 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。此外，此要求适用于托管节点组和自行管理的节点。

> **Note**
>
> 您不能使用创建任何集群时使用的相同角色。

必须先使用以下权限创建 IAM 角色，然后才能创建节点：

- kubelet 描述 VPC 中 Amazon EC2 资源的权限，例如 AmazonEKSWorkerNodePolicy 策略提供的权限。该策略还为 Amazon EKS 容器组身份代理提供权限。
- kubelet 使用来自 Amazon Elastic Container Registry（Amazon ECR）的容器映像的权限，例如 AmazonEC2ContainerRegistryReadOnly 策略提供的权限。使用来自 Amazon Elastic Container Registry（Amazon ECR）的容器映像的权限是必要条件，因为用于联网的内置附加组件运行的 Pod 使用来自 Amazon ECR 的容器映像。
- （可选）Amazon EKS 容器组身份代理使用 eks-auth:AssumeRoleForPodIdentity 操作检索容器组凭证的权限。如果您不使用 AmazonEKSWorkerNodePolicy，那么除了使用 EKS 容器组身份的 EC2 权限外，您还必须提供此权限。
- （可选）如果您不使用 IRSA 或 EKS 容器组身份向 VPC CNI 容器组授予权限，则必须为实例角色的 VPC CNI 提供权限。您可以使用 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（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 配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户（IRSA）。

> **Note**
>
> 在 2023 年 10 月 3 日之前，每个托管节点组的 IAM 角色上需要有 AmazonEKSWorkerNodePolicy 和 AmazonEC2ContainerRegistryReadOnly。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。

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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 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. 选择权限。
5. 确保将 AmazonEKSWorkerNodePolicy 和 AmazonEC2ContainerRegistryReadOnly 托管策略附加到此角色，或者使用最低权限附加自定义策略。

   > **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（IRSA）。

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 AWS Management Console 或 AWS CLI 创建节点 IAM 角色。

AWS Management Console

在 IAM 控制台中创建您的Amazon EKS 节点角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 AWS service（服务）。
   b. 在 Use case（使用案例）下，选择 EC2。
   c. 选择下一步。
5. 在添加权限页面上，附加自定义策略或执行以下操作：
   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（IRSA）。

f. 选择下一步。

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. 在添加标签（可选）下，将标签作为键值对附加，以将元数据添加到角色。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 资源。

d. 选择创建角色。

AWS 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（IRSA）。

   - 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 AWS 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 AWS Fargate 时，在 Fargate 基础设施上运行的组件必须代表您调用 AWS 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 AWS 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 AWS 服务的权限，您必须使用 服务账户的 IAM 角色。

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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 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. 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 AWS 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 AWS 区域。如果您希望在您的账户中在所有 AWS 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 AWS Management Console 或 AWS CLI 来创建它。

AWS Management Console

要使用 AWS Management Console 创建 AWS FargatePod 执行角色

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 AWS service（服务）。

b. 从其他 AWS 服务 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. 在添加标签（可选）下，将标签作为键值对附加，以将元数据添加到角色。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 资源。

c. 选择创建角色。

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 AWS 区域。如果您希望在您的账户中在所有 AWS 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）。

AWS CLI

要使用 AWS CLI 创建 AWS FargatePod 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 AWS 区域。如果您希望在您的账户中在所有 AWS 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 AWS Management Console 中查看。要连接 Kubernetes 集群，创建 IAM 角色。

检查现有 EKS 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 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 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. 选择权限。
5. 确保将 AmazonEKSConnectorAgentPolicy 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 connector 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 AWS Management Console 或 AWS CloudFormation 创建 connector 代理角色。

AWS 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
```

AWS CloudFormation

要用 AWS CloudFormation 创建 Amazon EKS Connector 代理角色

1. 将以下 AWS 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。

```
---
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 AWS #
#.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

2. 打开 AWS 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 配置堆栈选项 页面上，请选择 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AWS托管策略

AWS 托管策略是由 AWS 创建和管理的独立策略。AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，AWS 托管式策略可能不会为您的特定使用场景授予最低权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新在 AWS 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 AWS 服务启动或新的 API 操作可用于现有服务时，AWS 最有可能更新 AWS 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS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，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，请参阅使用 Amazon VPC CNI plugin for Kubernetes Amazon EKS 附加组件 和配置 Amazon VPC CNI plugin for Kubernetes 将 IAM 角色用于服务账户（IRSA）。

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:*NetworkInterface** 和 **ec2:*PrivateIpAddresses** – 允许 Amazon VPC CNI 插件执行操作，例如为 Pods 预调配弹性网络接口和 IP 地址，以便为在 Amazon EKS 中运行的应用程序提供联网。
- **ec2** 读取操作 – 允许 Amazon VPC CNI 插件执行操作，例如描述实例和子网，以查看您的 Amazon VPC 子网中的免费 IP 地址数量。VPC CNI 可以使用每个子网中的免费 IP 地址来选择空闲 IP 地址最多的子网，以便在创建弹性网络接口时使用。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKS_CNI_Policy。

AWS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。由 Amazon EKS 托管的 Kubernetes 集群会代表您调用其他 AWS 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 AWS 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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSClusterPolicy。

AWS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 和AWS Fargate 配置文件。

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 AWS 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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSFargatePodExecutionRolePolicy。

AWS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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSForFargateServiceRolePolicy。

AWS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 AWSServiceRoleforAmazonEKS 服务相关角色。服务相关角色附加了 AWS托管策略：AmazonEKSServiceRolePolicy。

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 服务相关角色。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSServicePolicy。

AWS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。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 AWSServiceRoleforAmazonEKS 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AWS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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSServiceRolePolicy。

AWS托管策略：AmazonEKSVPCResourceController

您可以将 `AmazonEKSVPCResourceController` 策略附加到 IAM 身份。如果您使用适用于 Pods 的安全组，您必须将此策略附加到您的 Amazon EKS 集群 IAM 角色，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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSVPCResourceController。

AWS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。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。
- **eks-auth:AssumeRoleForPodIdentity**：允许检索节点上 EKS 工作负载的凭证。需要执行此操作 EKS 容器组身份才能正常运行。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSWorkerNodePolicy。

AWS托管策略：AWSServiceRoleForAmazonEKSNodegroup

您不能将 AWSServiceRoleForAmazonEKSNodegroup 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。

此策略授予 AWSServiceRoleForAmazonEKSNodegroup 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。这些权限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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AWSServiceRoleForAmazonEKSNodegroup。

AWS 托管策略：AmazonEBSCSIDriverPolicy

AmazonEBSCSIDriverPolicy 策略允许 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。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEBSCSIDriverServiceRolePolicy。

AWS 托管式策略：AmazonEFSCSIDriverPolicy

AmazonEFSCSIDriverPolicy 策略允许 Amazon EFS 容器存储接口（CSI）代表您创建和删除接入点。该策略还将向 Amazon EFS CSI 驱动程序授予列出您的接入点文件系统、挂载目标和 Amazon EC2 可用区的权限。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEFSCSIDriverServiceRolePolicy。

AWS 托管策略：AmazonEKSLocalOutpostClusterPolicy

您可以将此策略附加到 IAM 实体。在创建本地集群之前，您必须将此策略附加到您的集群角色。由 Amazon EKS 托管的 Kubernetes 集群会代表您调用其他 AWS 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。

AmazonEKSLocalOutpostClusterPolicy 包含以下权限：

- **ec2** – Amazon EC2 实例作为控制面板实例成功加入集群所需的权限。
- **ssm** - 允许通向控制面板实例的 Amazon EC2 Systems Manager 连接，Amazon EKS 使用该连接与您账户中的本地集群通信并对其进行管理。
- **logs** - 允许实例将日志推送到 Amazon CloudWatch。
- **secretsmanager** - 允许实例安全地从 AWS Secrets Manager 中获取和删除控制面板实例的引导数据。
- **ecr** - 允许 Pods 和在控制面板实例上运行的容器拉取存储在 Amazon Elastic Container Registry 中的容器映像。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSLocalOutpostClusterPolicy。

AWS 托管策略：AmazonEKSLocalOutpostServiceRolePolicy

您不能将此策略附加到您的 IAM 实体。当您使用具有 `iam:CreateServiceLinkedRole` 权限的 IAM 主体创建集群时，Amazon EKS 将自动为您创建 `AWSServiceRoleforAmazonEKSLocalOutpost` 服务相关角色，并将此策略附加到该角色。此策略允许该服务相关角色代表您为本地集群调用 AWS 服务。

`AmazonEKSLocalOutpostServiceRolePolicy` 包含以下权限：

- **`ec2`** - 允许 Amazon EKS 使用安全、网络和其他资源，成功启动和管理您的账户中的控制面板实例。
- **`ssm`** - 允许通向控制面板实例的 Amazon EC2 Systems Manager 连接，Amazon EKS 使用该连接与您账户中的本地集群通信并对其进行管理。
- **`iam`** - 允许 Amazon EKS 管理与控制面板实例关联的实例配置文件。
- **`secretsmanager`** - 允许 Amazon EKS 将控制面板实例的引导数据放入 AWS Secrets Manager，以便能在实例引导期间安全地引用它。
- **`outposts`** - 允许 Amazon EKS 从您的账户中获取 Outpost 信息，以便在 Outpost 中成功启动本地集群。

要查看 JSON 策略文档的最新版本，请参阅《AWS 托管式策略参考指南》中的 AmazonEKSLocalOutpostServiceRolePolicy。

AWS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AWS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	描述	日期
AmazonEKS_CNI_Policy – 更新到现有策略	Amazon EKS 添加了新的 `ec2:DescribeSubnets` 权限，允许 Amazon VPC CNI plugin for Kubernetes 查看您的 Amazon VPC 子网中的免费 IP 地址数量。	2024 年 3 月 4 日

更改	描述	日期
	VPC CNI 可以使用每个子网中的免费 IP 地址来选择空闲 IP 地址最多的子网，以便在创建弹性网络接口时使用。	
AmazonEKSWorkerNodePolicy：更新现有策略	Amazon EKS 添加了新的权限以允许 EKS 容器组身份。 Amazon EKS 容器组身份代理使用节点角色。	2023 年 11 月 26 日
推出了 AmazonEFSCSIDriverPolicy。	AWS 引入了 `AmazonEFSCSIDriverPolicy` 。	2023 年 7 月 26 日
已将权限添加至 AmazonEKSClusterPolicy。	增加了 `ec2:DescribeAvailabilityZones` 权限以允许 Amazon EKS 在创建负载均衡器时在子网自动发现期间获取可用区详细信息。	2023 年 2 月 7 日
更新了 AmazonEBSCSIDriverPolicy 中的策略条件。	删除了 `StringLike` 键字段中带有通配符的无效策略条件。还将一个新条件 `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` 添加到 `ec2:DeleteVolume` ，以允许 EBS CSI 驱动程序删除由树内插件创建的卷。	2022 年 11 月 17 日

更改	描述	日期
添加了对 AmazonEKSLocalOutpostServiceRolePolicy 的权限。	添加了 ec2:DescribeVPCAttribute 、ec2:GetConsoleOutput 和 ec2:DescribeSecret 以实现更好的先决条件验证和托管式生命周期控制。还添加了 ec2:DescribePlacementGroups 、"arn:aws:ec2:*:*:placement-group/*" 和 ec2:RunInstances 以支持控制面板 Amazon EC2 实例在 Outposts 上的置放控制。	2022 年 10 月 24 日
更新了 AmazonEKSLocalOutpostClusterPolicy 中的 Amazon Elastic Container Registry 权限。	将操作 ecr:GetDownloadUrlForLayer 从所有资源部分移至限定范围部分。添加了资源 arn:aws:ecr:*:*:repository/eks/* 。删除了资源 arn:aws:ecr:*:*:repository/eks/eks-certificates-controller-public 。此资源涵盖在增加的 arn:aws:ecr:*:*:repository/eks/* 资源中。	2022 年 10 月 20 日
将权限添加到了 AmazonEKSLocalOutpostClusterPolicy。	添加了 arn:aws:ecr:*:*:repository/kubelet-config-updater Amazon Elastic Container Registry 存储库，以便集群控制面板实例能够更新某些 kubelet 参数。	2022 年 8 月 31 日
引入了 AmazonEKSLocalOutpostClusterPolicy。	AWS 引入了 AmazonEKSLocalOutpostClusterPolicy 。	2022 年 8 月 24 日
引入了 AmazonEKSLocalOutpostServiceRolePolicy。	AWS 引入了 AmazonEKSLocalOutpostServiceRolePolicy 。	2022 年 8 月 23 日

更改	描述	日期
引入的 AmazonEBSCSIDriverPolicy。	AWS 引入了 AmazonEBSCSIDriverPolicy 。	2022 年 4 月 22 日
已添加权限到 AmazonEKSWorkerNodePolicy。	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。	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。	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AWS托管策略的更改。	2021 年 6 月 17 日

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 AWS API 操作时收到 AccessDeniedException，则表明您使用的 IAM 主体凭证不具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 IAM 主体提供 Amazon EKS 管理员权限，请参阅Amazon EKS 基于身份的策略示例。

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。

无法在计算选项卡上看到节点，或在资源选项卡上看到任何内容，您将在 AWS 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。确保您将 AWS Management Console 与其结合使用的 IAM 主体用户具有必要的权限。有关更多信息，请参阅 所需的权限。

aws-auth **`ConfigMap`** 不授予对集群的访问权限

AWS 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 AWS 服务 允许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 AWS 管理员。您的管理员是提供登录凭证的人。

我希望允许我的AWS账户以外的人访问我的 Amazon EKS 资源

您可以创建一个角色，以便其他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，可以担任角色。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。
- 要了解如何为您拥有的 AWS 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 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 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 AWS 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存取之间的差别，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 AWS 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 AWS 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.us-west-2.amazonaws.com`。您的应用程序应内置冗余功能，以便在该 AWS 区域 的服务出现故障时选择其他 AWS 区域。有关更多信息，请参阅 IAM 用户指南中的在 AWS 区域中管理 AWS 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` 版和更高版本集群默认使用区域端点。有关更多信息，请参阅 为服务账户配置 AWS Security Token Service 端点。

默认 Amazon EKS 创建的 Kubernetes 角色和用户

创建 Kubernetes 集群时，会在该集群上创建多个默认 Kubernetes 身份，以便 Kubernetes 正常运行。Amazon EKS 会为其每个默认组件创建 Kubernetes 身份。这些身份为集群组件提供 Kubernetes 基于角色的授权控制（RBAC）。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

当您向集群安装可选附加组件时，可能会向您的集群添加其他 Kubernetes 身份。有关本主题未涉及身份的更多信息，请参阅附加组件文档。

您可以使用 AWS Management Console 或 `kubectl` 命令行工具查看 Amazon EKS 在集群上创建的 Kubernetes 身份列表。所有用户身份都会出现在 kube 审计日志中，可通过 Amazon CloudWatch 向你提供。

AWS Management Console

先决条件

您使用的 IAM 主体必须拥有 所需的权限 中描述的权限。

要使用AWS Management Console查看 Amazon EKS 创建的身份

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，选择包含要查看的身份的集群。
3. 选择资源选项卡。
4. 在 Resource types（资源类型）下，选择 Authorization（授权）。
5. 选择 ClusterRoles、ClusterRoleBindings、Roles 或 RoleBindings。所有以 eks 为前缀的资源均由 Amazon EKS 创建。Amazon EKS 创建的其他身份资源包括：
   - ClusterRole 和名为 aws-node 的 ClusterRoleBinding。aws-node 资源支持 Amazon VPC CNI plugin for Kubernetes，Amazon EKS 会将其安装在所有集群上。
   - 名为 vpc-resource-controller-role 的 ClusterRole 和名为 vpc-resource-controller-rolebinding 的 ClusterRoleBinding。这些资源支持 Amazon VPC 资源控制器，Amazon EKS 会将其安装在所有集群上。

   除了控制台中的资源外，即使以下特殊用户身份在集群配置中不可见，但它们仍存在于您的集群上：

   - **`eks:cluster-bootstrap`**：在集群引导期间用于 `kubectl` 操作。
   - **`eks:support-engineer`**：用于集群管理操作。
6. 选择特定资源以查看有关该资源的详细信息。默认情况下，信息在 Structured view（结构化视图）中显示。在详细信息页面的右上角，您可以选择 Raw view（原始视图）以查看该资源的所有信息。

Kubectl

先决条件

您用于列出集群上的 Kubernetes 资源的实体（AWS Identity and Access Management（IAM）或 OpenID Connect（OIDC））必须由 IAM 或您的 OIDC 身份提供商进行身份验证。必须向实体授予权限，才能为您希望该实体使用的集群上的 `Role`、`ClusterRole`、`RoleBinding` 和 `ClusterRoleBinding` 资源使用 Kubernetes `get` 和 `list` 动词。有关向 IAM 实体授予集群访问权限的更多信息，请参阅the section called “授予对 Kubernetes API 的访问权限”。有关向经过您自己的 OIDC 提供商身份验证的实体授予集群访问权限的更多信息，请参阅通过 OpenID Connect 身份提供商对集群的用户进行身份验证。

要使用 **`kubectl`** 查看 Amazon EKS 创建的身份

为要查看的资源类型运行命令。所有以 eks 为前缀的返回资源均由 Amazon EKS 创建。除命令输出中返回的资源，即使以下特殊用户身份在集群配置中不可见，但它们仍存在于您的集群上：

- **`eks:cluster-bootstrap`**：在集群引导期间用于 `kubectl` 操作。
- **`eks:support-engineer`**：用于集群管理操作。

ClusterRoles：`ClusterRoles` 范围限定为您的集群，因此授予角色的任何权限都适用于集群上任何 Kubernetes 命名空间中的资源。

以下命令返回 Amazon EKS 在您的集群上创建的所有 Kubernetes `ClusterRoles`。

```
kubectl get clusterroles | grep eks
```

除了输出中返回的 `ClusterRoles`（具有前缀）外，还存在以下 `ClusterRoles`。

- **`aws-node`**：此 `ClusterRole` 支持 Amazon VPC CNI plugin for Kubernetes，Amazon EKS 会将其安装在所有集群上。
- **`vpc-resource-controller-role`**：此 `ClusterRole` 支持 Amazon VPC 资源控制器，Amazon EKS 会将其安装在所有集群上。

要查看 `ClusterRole` 的规范，请将以下命令中的 *`eks:k8s-metrics`* 替换为上一条命令输出中返回的 `ClusterRole`。以下示例返回 *`eks:k8s-metrics`* `ClusterRole` 的规范。

```
kubectl describe clusterrole eks:k8s-metrics
```

示例输出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings：ClusterRoleBindings 范围限定为您的集群。

以下命令返回 Amazon EKS 在您的集群上创建的所有 Kubernetes ClusterRoleBindings。

```
kubectl get clusterrolebindings | grep eks
```

除了输出中返回的 ClusterRoleBindings 外，还存在以下 ClusterRoleBindings。

- **aws-node**：此 ClusterRoleBinding 支持 Amazon VPC CNI plugin for Kubernetes，Amazon EKS 会将其安装在所有集群上。
- **vpc-resource-controller-rolebinding**：此 ClusterRoleBinding 支持 Amazon VPC 资源控制器，Amazon EKS 会将其安装在所有集群上。

要查看 ClusterRoleBinding 的规范，请将以下命令中的 *eks:k8s-metrics* 替换为上一条命令输出中返回的 ClusterRoleBinding。以下示例返回 *eks:k8s-metrics* ClusterRoleBinding 的规范。

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

示例输出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
```

```
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Roles：Roles 范围限定为 Kubernetes 命名空间。Amazon EKS 创建的所有 Roles 范围限定为 kube-system 命名空间。

以下命令返回 Amazon EKS 在您的集群上创建的所有 Kubernetes Roles。

```
kubectl get roles -n kube-system | grep eks
```

要查看 Role 的规范，请将以下命令中的 *eks:k8s-metrics* 替换为上一条命令输出中返回的 Role 名称。以下示例返回 *eks:k8s-metrics* Role 的规范。

```
kubectl describe role eks:k8s-metrics -n kube-system
```

示例输出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names              Verbs
  ---------         -----------------  --------------              -----
  daemonsets.apps   []                 [aws-node]                  [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings：RoleBindings 范围限定为 Kubernetes 命名空间。Amazon EKS 创建的所有 RoleBindings 范围限定为 kube-system 命名空间。

以下命令返回 Amazon EKS 在您的集群上创建的所有 Kubernetes RoleBindings。

```
kubectl get rolebindings -n kube-system | grep eks
```

要查看 RoleBinding 的规范，请将以下命令中的 *eks:k8s-metrics* 替换为上一条命令输出中返回的 RoleBinding。以下示例返回 *eks:k8s-metrics* RoleBinding 的规范。

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

示例输出如下。

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Amazon Elastic Kubernetes Service 的合规性验证

要了解某个 AWS 服务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 AWS 服务，然后选择您感兴趣的合规性计划。有关常规信息，请参阅AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告、。

您使用 AWS 服务 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。AWS 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 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署以安全性和合规性为重点的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计– 该白皮书介绍了公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。

> **Note**
>
> 并非所有 AWS 服务都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 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。
- AWS 客户合规指南：从合规角度了解责任共担模式。这些指南总结了保护 AWS 服务 的最佳实践，并将指南映射到跨多个框架的安全控制，包括美国国家标准与技术研究院（NIST）、支付卡行业安全标准委员会（PCI）和国际标准化组织（ISO）。

- AWS Config 开发人员指南中的使用规则评估资源 - 此 AWS Config 服务评测您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – 此 AWS 服务 向您提供 AWS 中安全状态的全面视图。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty – 该 AWS 服务 通过监控您的环境中是否存在可疑和恶意活动，来检测您的 AWS 账户、工作负载、容器和数据面临的潜在威胁。GuardDuty 可以通过满足某些合规性框架规定的入侵检测要求，来协助您满足各种合规性要求，如 PCI DSS。
- AWS Audit Manager – 此 AWS 服务 可帮助您持续审核您的 AWS 使用情况，以简化管理风险以及与相关法规和行业标准的合规性的方式。

Amazon EKS 中的恢复能力

AWS全球基础设施围绕AWS 区域和可用区构建。AWS 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 AWS 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 AWS 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 AWS 区域内的可用区重新启动它们。
- 利用 AWS 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 AWS 区域 和可用区的更多信息，请参阅 AWS 全球基础设施。

Amazon EKS 中的基础设施安全性

作为一项托管式服务，Amazon Elastic Kubernetes Service 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实操设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用AWS发布的 API 调用通过网络访问 Amazon EKS。客户端必须支持以下内容：

- 传输层安全性协议 (TLS) 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。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 AWS Security Token Service（AWS 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 Pods。

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。

如果使用 开始使用 Amazon EKS 演练中提供的 AWS 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。

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 AWS 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。

您可以使用 Amazon VPC CNI 或第三方工具（如项目 Calico）实施 Kubernetes 网络策略。有关将 Amazon VPC CNI 用于网络策略的更多信息，请参阅 为 Kubernetes 网络策略配置集群。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。下方列出了供您分析 EKS 集群安全配置的资源、用于检查漏洞的资源，以及可为您执行该分析的 AWS 服务的集成。

适用于 Amazon EKS 的互联网安全中心（CIS）基准

Center for Internet Security（CIS）Kubernetes Benchmark 提供了关于 Amazon EKS 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

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。

您可以将 Amazon EKS 集群更新到更新的 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。

操作系统漏洞列表

AL2023 漏洞列表

在 Amazon Linux 安全中心跟踪 Amazon Linux 2023 的安全或隐私事件，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。

Amazon Linux 2 漏洞列表

在 Amazon Linux 安全中心跟踪 Amazon Linux 2 的安全或隐私事件，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。

使用 Amazon Inspector 进行节点检测

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。

使用 Amazon GuardDuty 进行集群和节点检测

Amazon GuardDuty 是一项威胁检测服务，有助于保护您的账户、容器、工作负载和 AWS 环境中的数据。除其他功能外，GuardDuty 还提供以下两项功能，用于检测 EKS 集群面临的潜在威胁：EKS 保护和运行时监控。

有关更多信息，请参阅 使用 Amazon GuardDuty 检测威胁。

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/

容器组（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**
>
> `PodSecurityPolicy`（PSP）在 Kubernetes 版本 `1.21` 中已弃用，且已在 Kubernetes `1.25` 中删除。PSPs 将被容器组（pod）安全准入（PSA）取代，其是一种内置的准入控制器，可实施容器组（pod）安全标准（PSS）中列出的安全控制。PSA 和 PSS 都处于测试版功能状态，默认情况下在 Amazon EKS 中处于启用状态。为了解决 `1.25` 版中的 PSP 删除问题，我们建议您在 Amazon EKS 中实施 PSS。有关更多信息，请参阅 AWS 博客上的 Implementing Pod Security Standards in Amazon EKS（在 Amazon EKS 中实施容器组（pod）安全标准）。

Amazon EKS 默认 Pod 安全策略

带 Kubernetes 版本 1.13 或更高版本的 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
```

示例输出如下。

```
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
```

示例输出如下。

```
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 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 Pod 安全策略分配给用于 Daemonset 部署的 `aws-node` Pods 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 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
```

容器组（pod）安全策略 (PSP) 移除常见问题

PodSecurityPolicy 已在 Kubernetes1.21 中弃用，并且已在 Kubernetes1.25 中移除。如果您在集群中使用 PodSecurityPolicy，则必须先迁移到内置的 Kubernetes 容器组（pod）安全标准 (PSS) 或策略即代码解决方案，然后再将集群升级到版本 **1.25**，以避免工作负载中断。选择任何常见问题以了解更多信息。

什么是 PSP？

PodSecurityPolicy 是一个内置的准入控制器，集群管理员可以用它控制 Pod 规范的安全敏感方面。如果某个 Pod 满足其 PSP 的要求，则该 Pod 会像往常一样被准许进入集群。如果某个 Pod 不符合 PSP 要求，则该 Pod 会被拒绝并且无法运行。

PSP 移除是特定于 Amazon EKS 的还是正在上游 Kubernetes 中移除？

这是 Kubernetes 项目的上游更改，而不是 Amazon EKS 中的更改。PSP 已在 Kubernetes 1.21 中弃用并在 Kubernetes 1.25 中移除。Kubernetes 社区发现了严重的 PSP 可用性问题。其中包括意外授予比预期更广泛的权限，以及难以检查在给定情况下哪个 PSPs 适用。如果不进行重大更改，则无法解决这些问题。这是 Kubernetes 社区决定移除 PSP 的主要原因。

我如何检查我是否在 Amazon EKS 集群中使用 PSPs？

要确定自己是否在集群中使用 PSPs，您可以运行以下命令：

```
kubectl get psp
```

要查看您的集群中的 PSPs 影响的 Pods，请运行以下命令。此命令输出 Pod 名称、命名空间和 PSPs：

```
kubectl get pod -A -o jsonpath='{range.items[?(@.metadata.annotations.kubernetes
\.io/psp)]}{.metadata.name}{"\t"}{.metadata.namespace}{"\t"}
{.metadata.annotations.kubernetes\.io/psp}{"\n"}'
```

如果我在我的 Amazon EKS 集群中使用 PSPs，我能做什么？

在将集群升级到 1.25 之前，您必须将您的 PSPs 迁移到以下任一备选方案：

- Kubernetes PSS.

- 来自 Kubernetes 环境的策略即代码解决方案。

为了应对 PSP 弃用以及从一开始就控制 Pod 安全的持续需求，Kubernetes 社区创建了一个包含（PSS）和容器组（pod）安全准入（PSA）的内置解决方案。PSA Webhook 实施 PSS 中定义的控件。

您可以查看 EKS 最佳实践指南中的将 PSPs 迁移到内置的 PSS 的最佳实践。我们还建议您阅读关于在 Amazon EKS 中实施容器组（pod）安全标准的博客。其他参考资料包括从 PodSecurityPolicy 迁移到内置的 PodSecurity 准入控制器以及将 PodSecurityPolicies 映射到容器组（pod）安全标准。

策略即代码解决方案提供防护机制来指导集群用户，并通过规定的自动控件来防止不需要的行为。策略即代码解决方案通常使用 Kubernetes 动态准入控制器通过 Webhook 调用截获 Kubernetes API 服务器请求流。策略即代码解决方案根据以代码形式编写和存储的策略对请求有效负载进行更改和验证。

有几种开源策略即代码解决方案可供 Kubernetes 使用。要查看将 PSPs 迁移到策略即代码解决方案的最佳实践，请参阅 GitHub 上容器组（pod）安全页面的策略即代码部分。

我在我的集群中看到一个称为 **eks.privileged** 的 PSP。这是什么，我能做些什么？

带 Kubernetes 版本 `1.13` 或更高版本的 Amazon EKS 集群具有名为 `eks.privileged` 的默认 PSP。此策略在 `1.24` 及更早版本的集群中创建。它不在 `1.25` 及更高版本的集群中使用。Amazon EKS 会自动将此 PSP 迁移到基于 PSS 的强制实施中。您无需执行任何操作。

当我将集群更新到 **1.25** 版本时，Amazon EKS 是否会对存在于现有集群中的 PSPs 进行更改？

不会。除了 `eks.privileged`（由 Amazon EKS 创建的 PSP）之外，升级到 `1.25` 时，不会对集群中的其他 PSPs 进行任何更改。

如果我尚未从 PSP 中迁移出，Amazon EKS 是否会阻止集群更新到版本 **1.25**？

不会。如果您尚未迁移出 PSP，Amazon EKS 将不会阻止集群更新到版本 `1.25`。

如果我在将集群更新到版本 **1.25** 之前忘记将自己的 PSPs 迁移到 PSS/PSA 或策略即代码解决方案，该怎么办？ 我是否可以在更新集群后迁移？

当包含 PSP 的集群升级到 Kubernetes 版本 `1.25` 时，API 服务器不会识别 `1.25` 中的 PSP 资源。这可能会导致 Pods 获得错误的安全范围。有关含义的详尽列表，请参阅从 PodSecurityPolicy 迁移到内置的 PodSecurity Admission 准入控制器。

这一变化如何影响 Windows 工作负载的容器组（pod）安全？

我们预计不会对 Windows 工作负载产生任何特定影响。PodSecurityContext 在适用于 Windows Pods 的 `PodSpec v1` API 中有一个名为 `windowsOptions` 的字段。这会使用 Kubernetes `1.25` 中的 PSS。有关强制执行 Windows 工作负载的 PSS 的更多信息和最佳实践，请参阅 EKS 最佳实践指南和 Kubernetes 文档。

将 Kubernetes 与 AWS 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 AWS 密钥和配置提供程序（ASCP）。

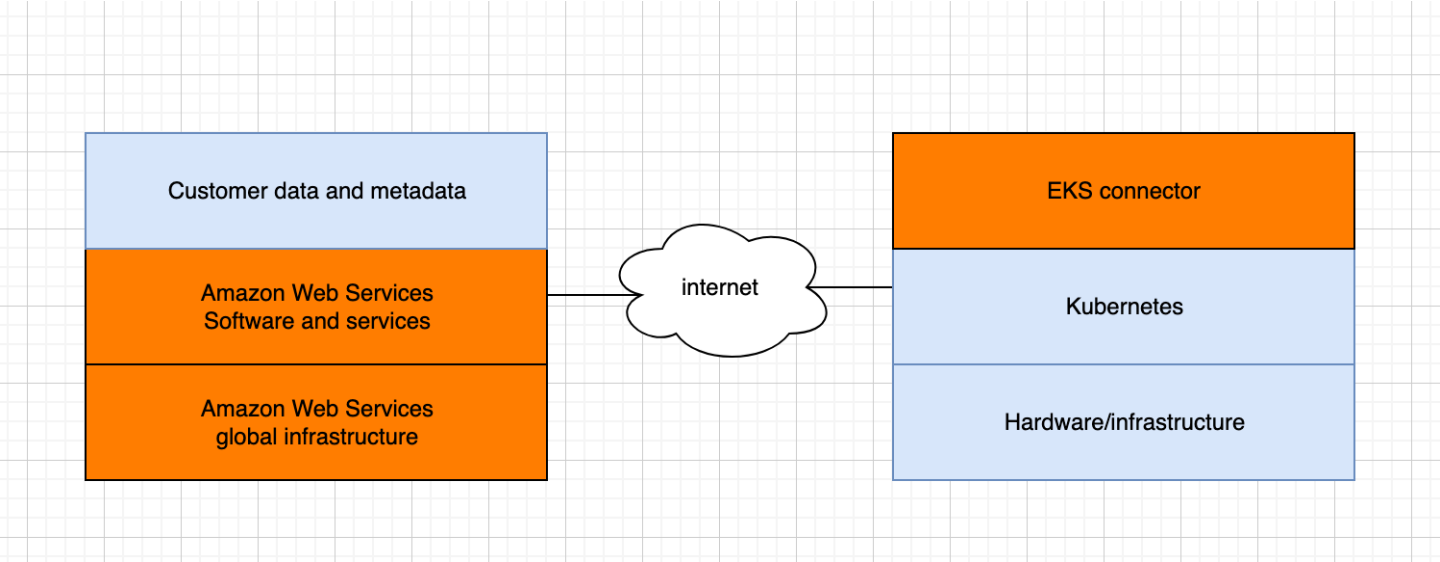
使用 ASCP，您可以在 Secrets Manager 中存储并管理密钥，然后通过 Amazon EKS 上运行的工作负载检索。您可以使用 IAM 角色和策略限制集群中特定 Kubernetes Pods 钥访问权限。ASCP 检索 Pod 标识并交换 IAM 角色的身份。ASCP 担任 Pod 的 IAM 角色，然后可以从授权该角色的 Secrets Manager 中检索密钥。

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 AWS 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

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 AWS 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 AWS 责任，蓝色代表客户责任：



本主题介绍了连接的集群在 AWS 外部时责任模型的差异。

AWS 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 AWS 通信的开源组件。
- 维护连接的 Kubernetes 集群和 AWS 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 AWS 的 IAM 主体访问权限。有关说明，请参阅 向 IAM 主体授予查看集群上的 Kubernetes 资源的访问权限。
- 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 AWS 账户安全（例如，通过保护您的根用户凭证安全）。

查看 Kubernetes 资源

您可以使用 AWS Management Console 查看部署到您的集群的 Kubernetes 资源。您无法使用 AWS CLI 或 `eksctl` 查看 Kubernetes 资源。要使用命令行工具查看 Kubernetes 资源，请使用 `kubectl`。

先决条件

要查看 AWS Management Console 中的计算选项卡上的资源选项卡和节点部分，您使用的 IAM 主体必须具有特定的 IAM 和 Kubernetes 权限。有关更多信息，请参阅 所需的权限。

使用 AWS 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. 选择资源选项卡。
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，例如工作负载组中的部署。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。
   - 选择集群组，然后选择节点资源类型。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。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。

所需的权限

要查看 AWS Management Console 中的计算选项卡上的资源选项卡和节点部分，您使用的 IAM 主体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 IAM 主体。

1. 确保向您正在使用的 IAM 主体分配 `eks:AccessKubernetesApi` 以及查看 Kubernetes 资源所需的其他 IAM 权限。有关如何为 IAM 主体编辑权限的更多信息，请参阅《IAM 用户指南》中的控制主体的访问权限。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）。

   以下示例策略包括主体查看账户中所有集群的 Kubernetes 资源所需的权限。将 *111122223333* 替换为您的AWS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/*"
        }
    ]
}
```

要查看已连接集群中的节点，Amazon EKS 连接器 IAM 角色应能够模拟集群中的主体。这使得 Amazon EKS Connector 能够将主体映射到 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.us-west-2.amazonaws.com/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.us-west-2.amazonaws.com/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 https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
      ```

      ```
      curl -O https://s3.us-west-2.amazonaws.com/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 主体映射到 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 AWS CloudShell 上安装了 0.183.0 版或更高版本的 eksctl 命令行工具。要安装或更新 eksctl，请参阅 eksctl 文档中的 Installation。

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 AWS 区域。

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

示例输出如下。

```
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。

示例输出如下。

```
[...]
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。IAM 最佳实践建议您向角色而不是用户授予权限。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
```

示例输出如下。

```
[...]
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
```

示例输出如下。

```
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 中的更多信息，请参阅将 IAM 主体添加到 Amazon EKS 集群。

1. 打开 aws-auth ConfigMap 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 主体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。将 *111122223333* 替换为您的AWS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. 保存文件并退出文本编辑器。

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。您的 Amazon EKS 日志数据可以流式传输到 AWS 服务或合作伙伴工具来进行数据分析。AWS 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。您还可以使用 AWS 支持的开源解决方案来监控 Amazon EKS 基础设施。

在 Amazon EKS 控制台的左侧导航窗格中选择集群后，您可以通过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。

监控是保持 Amazon EKS 和您的 AWS 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 AWS 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，请确保您的监控计划可以解决以下问题。

- 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 AWS 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。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，以及基于角色的信息。有关更多信息，请参阅《AWS 规范性指导》中的 Amazon EKS 的日志记录和监控。

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。

> **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 AWS CloudTrail 集成，后者是在 Amazon EKS 中提供用户、角色或AWS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 AWS CloudTrail 记录 Amazon EKS API 调用。

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 指标。

要配置 Amazon CloudWatch 自定义日志的 Fluent Bit，请参阅《Amazon CloudWatch 用户指南》中的设置 Fluent Bit。

Amazon EKS 日志记录和监控工具

Amazon Web Services 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。

日志记录工具

领域	工具	描述	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

领域	工具	描述	设置
控制层面	AWS CloudTrail	它记录用户、角色或服务进行的 API 调用。	设置过程
AWS Fargate 实例的多个区域	AWS Fargate 日志路由器	对于 AWS Fargate 实例，它将日志流式传输到 AWS 服务或合作伙伴工具。将 AWS 用于 Fluent Bit。日志可以流式传输到其他 AWS 服务 或合作伙伴工具。	设置过程

监控工具

领域	工具	描述	设置
应用程序	CloudWatch Container Insights	CloudWatch Container Insights 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
应用程序	AWS 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 AWS 监控服务或合作伙伴。它可以通过 CloudWatc	设置过程

领域	工具	描述	设置
		h Container Insights 设置。	
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。	设置过程
应用程序	AWS 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
应用程序	Amazon CloudWatch Observability Operator	Amazon CloudWatch Observability Operator 收集指标、日志和跟踪数据。然后将其发送到 Amazon CloudWatch 和 AWS X-Ray。	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

Prometheus 指标

Prometheus 是一个用于抓取端点的监控和时间序列数据库。提供查询、聚合和存储收集的数据的功能。您还可以将其用于警报和警报聚合。本主题介绍了如何将 Prometheus 设置为托管或开源选项。一个常见用例为监控 Amazon EKS 控制面板指标。

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 AWS 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。

有关开启指标后如何使用 Prometheus 指标的详细信息，请参阅《Amazon Managed Service for Prometheus 用户指南》。

创建集群时开启 Prometheus 指标

> **Important**
>
> 适用于 Prometheus 的 Amazon 托管服务资源不在集群生命周期内，需要独立于集群进行维护。删除集群时，请务必同时删除所有适用的抓取器以停止适用的费用。有关更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的查找和删除抓取程序。

创建新集群时，您可以开启向 Prometheus 发送指标的选项。在 AWS Management Console 中，此选项位于创建新集群的配置可观测性步骤中。有关更多信息，请参阅 创建 Amazon EKS 集群。

Prometheus 通过一个名为抓取的拉取模型从集群中发现和收集指标。设置抓取程序以从您的集群基础设施和容器化应用程序中收集数据。

当您开启发送 Prometheus 指标的选项时，Amazon Managed Service for Prometheus 会提供一个完全托管的无代理抓取程序。使用以下高级配置选项，根据需要自定义默认抓取程序。

抓取程序别名

（可选）输入抓取程序的唯一别名。

目标位置

选择 Amazon Managed Service for Prometheus 工作区。工作区是专用于存储和查询 Prometheus 指标的逻辑空间。借助此工作区，您将能够查看有权访问该工作区账户的 Prometheus 指标。创建新工作区选项指示 Amazon EKS 使用您提供的工作区别名，来代表您创建工作区。您可以使用选择

现有工作区选项，从下拉列表中选择一个现有工作区。有关工作区的更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的管理工作区。

服务访问

本节总结了您在发送 Prometheus 指标时授予的权限：

- 允许 Amazon Managed Service for Prometheus 描述抓取的 Amazon EKS 集群
- 允许远程写入 Amazon Managed Prometheus 工作区

如果 `AmazonManagedScraperRole` 已存在，抓取程序会进行使用。选择 `AmazonManagedScraperRole` 链接以查看权限详细信息。如果 `AmazonManagedScraperRole` 尚不存在，请选择查看权限详细信息链接，以查看您通过发送 Prometheus 指标授予的特定权限。

子网

查看抓取程序将继承的子网。如果需要进行更改，请返回创建集群指定网络步骤。

安全组

查看抓取程序将继承的安全组。如果需要进行更改，请返回创建集群指定网络步骤。

抓取程序配置

根据需要修改 YAML 格式的抓取程序进行配置。为此，请使用表单或上传替换的 YAML 文件。有关更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的抓取程序配置。

Amazon Managed Service for Prometheus 指的是与集群一起创建的无代理抓取程序，作为 AWS 托管的收集器。有关 AWS 托管的收集器的更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的 AWS 托管的收集器。

> **Important**
>
> 您必须设置 `aws-auth ConfigMap` 才能向抓取程序提供集群内权限。有关更多信息，请参阅《Amazon Managed Service for Prometheus 用户指南》中的配置您的 Amazon EKS 集群。

查看 Prometheus 抓取程序详情

在创建集群并开启 Prometheus 指标选项的情况下，您可以查看 Prometheus 抓取程序详细信息。在 AWS Management Console 中查看集群时，请选择可观测性选项卡。一个显示集群抓取程序列表的表格，包括抓取程序 ID、别名、状态和创建日期等信息。

要查看有关抓取程序的更多详细信息，请选择抓取程序 ID 链接。例如，您可以查看抓取程序配置、Amazon 资源名称（ARN）、远程写入 URL 和网络信息。您可以使用抓取程序 ID 作为 Amazon Managed Service for Prometheus API 操作的输入，例如 `DescribeScraper` 和 `DeleteScraper`。您还可以使用 API 创建更多抓取程序。

有关使用 Prometheus API 的更多信息，请参阅 Amazon Managed Service for Prometheus API 参考。

使用 Helm 部署 Prometheus

或者，您可以使用 Helm V3 将 Prometheus 部署到您的集群。如果您已经安装 Helm，可以使用 `helm version` 命令检查您的版本。Helm 是 Kubernetes 集群的包管理器。有关 Helm 以及如何安装的更多信息，请参阅 将 Helm 与 Amazon EKS 结合使用。

在为您的 Amazon EKS 集群配置 Helm 后，您可以使用其通过以下步骤来部署 Prometheus。

要使用 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.persistence.storageClass="gp2" \
    --set 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 prometheus-community`，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
```

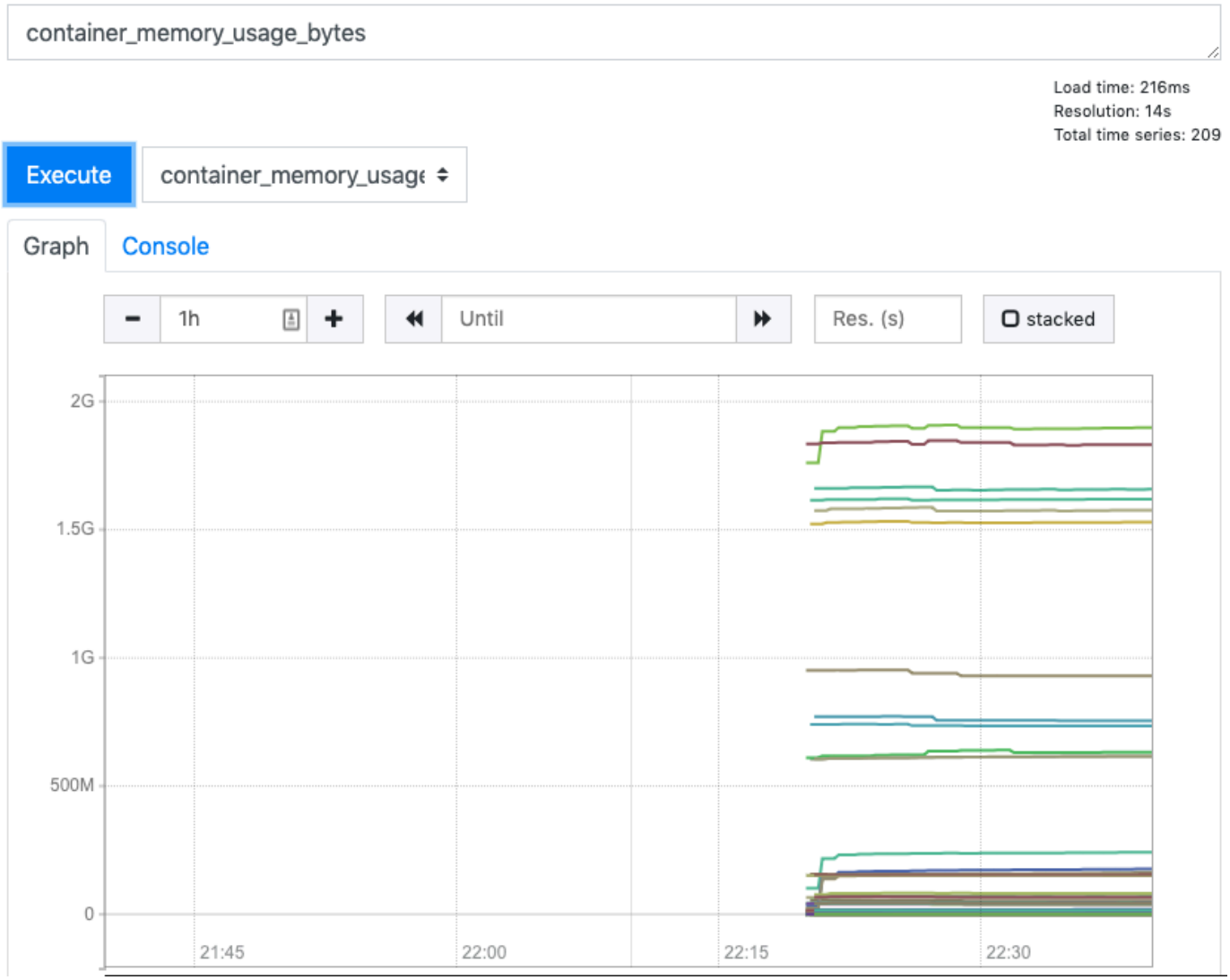
示例输出如下。

```
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
```

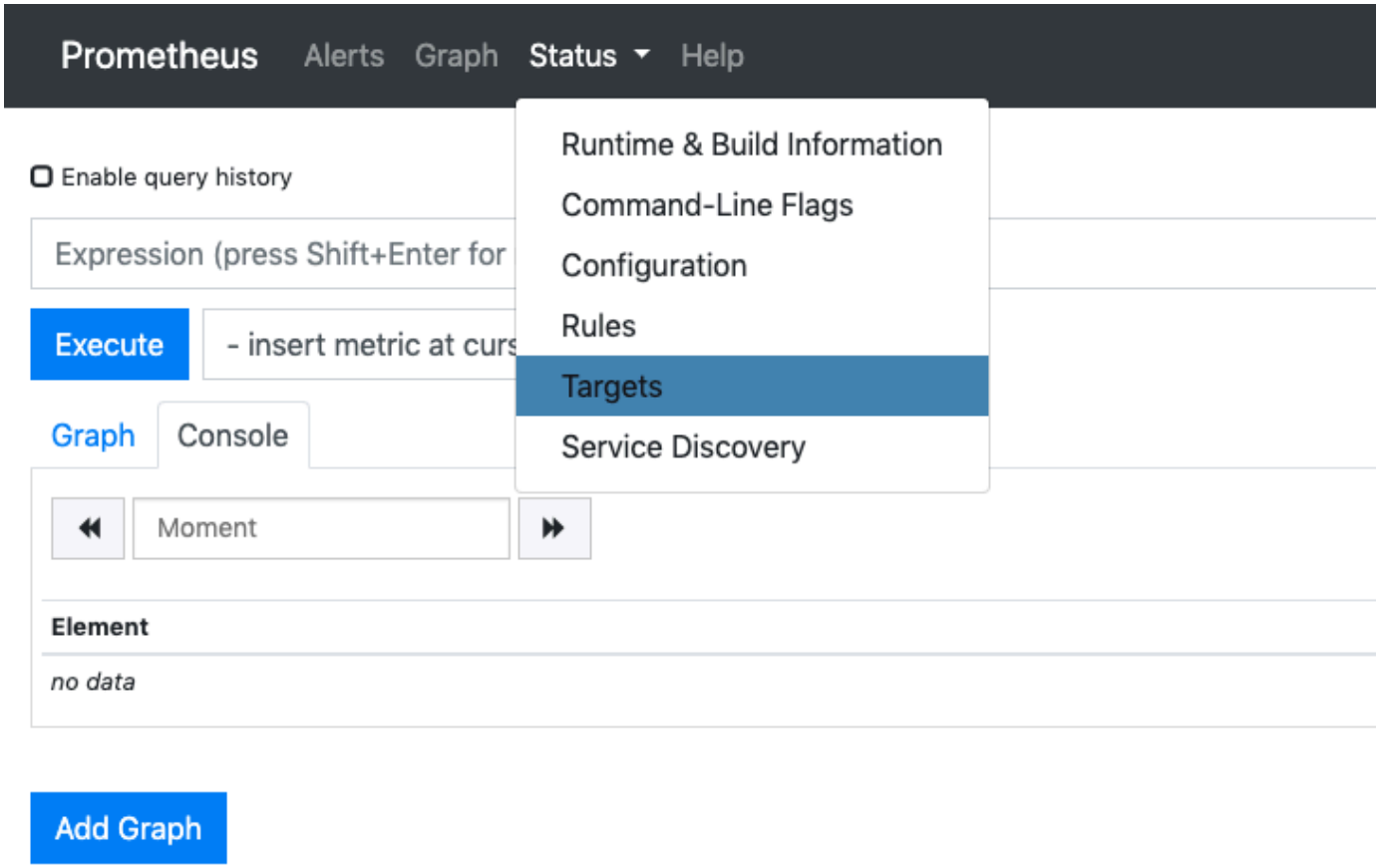
5. 使用 `kubectl` 将 Prometheus 控制台端口转发到本地计算机。

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. 将 Web 浏览器指向 `http://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。

查看控制面板原始指标

作为部署 Prometheus 的替代方案，Kubernetes API 服务器公开了许多以 Prometheus 格式表示的指标。这些指标对于监控和分析非常有用。这些指标通过引用 `/metrics` HTTP API 的指标端点在内部公开。与其他端点一样，此端点在 Amazon EKS 控制层面上公开。此端点主要用于查看特定指标。要分析一段时间内的指标，我们建议部署 Prometheus。

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
```

示例输出如下。

```
[...]
```

```
# 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。不同的指标按行列出，每行都包含指标名称、标签和值。

```
metric_name{"tag"="value"[,...]}
            value
```

适用于 Amazon CloudWatch 的 Amazon EKS 插件支持

Amazon CloudWatch Observability 收集实时日志、指标和跟踪数据。然后将其发送到 Amazon CloudWatch 和 AWS X-Ray。您可以安装此插件，以启用 CloudWatch Application Signals 和 CloudWatch Container Insights，从而增强 Amazon EKS 的可观测性。这有助于您监控基础设施和容器化应用程序的运行状况和性能。Amazon CloudWatch Observability Operator 旨在安装和配置必要的组件。

Amazon EKS 支持 Amazon CloudWatch Observability Operator 作为 Amazon EKS 插件。该插件允许集群中的 Linux 和 Windows Worker 节点上的 Container Insights。要启用 Windows 上的 Container Insights，Amazon EKS 插件版本必须为 `1.5.0` 或更高版本。目前，Amazon EKS Windows 不支持 CloudWatch Application Signals。

下面几个主题介绍了如何将 Amazon CloudWatch Observability Operator 用于 Amazon EKS 集群。

- 有关安装此插件的说明，请参阅《Amazon CloudWatch 用户指南》中的使用 CloudWatch Observability Amazon EKS 插件安装 CloudWatch 代理。

- 有关 CloudWatch Application Signals 的更多信息，请参阅 Application Signals。
- 有关 Container Insights 的更多信息，请参阅《Amazon CloudWatch 用户指南》中的使用 Container Insights。

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。有关更多信息，请参阅 Amazon CloudWatch 日志记录。

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 AWS Management Console、AWS 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AWS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。

API 服务器 (**`api`**)

集群的 API 服务器是公开 Kubernetes API 的控制面板组件。如果您在启动集群时或在启动之后不久启用 API 服务器日志，则这些日志包括用于启动 API 服务器的 API 服务器标志。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 审计日志提供影响集群的单个用户、管理员或系统组件的记录。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。

控制器管理器 (**controllerManager**)

控制器管理器用来管理 Kubernetes 附带的核心控制环路。有关更多信息，请参阅 Kubernetes 文档中的 kube-controller-manager。

计划程序 (**scheduler**)

计划程序组件用来管理在集群中运行 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。

为了更新控制面板日志记录配置，AmazonEKS 要求每个子网中最多有五个可用的 IP 地址。启用日志类型后，发送的日志具有详细程度级别 2。

AWS Management Console

使用 AWS Management Console 启用或禁用控制层面日志

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. 选择可观测性选项卡。
4. 在控制面板日志部分，选择管理日志记录。
5. 对于每个日志类型，选择开启还是关闭日志类型。默认情况下，每种日志类型都处于关闭状态。
6. 选择 Save changes（保存更改）以完成操作。

AWS CLI

使用 AWS CLI 启用或禁用控制层面日志

1. 使用以下命令查看您的 AWS CLI 版本。

```
aws --version
```

如果您的 AWS 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 AWS CLI，请参阅 AWS Command Line Interface 用户指南中的安装 AWS Command Line Interface。

2. 使用下面的 AWS 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}]}'
```

示例输出如下。

```
{
    "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
```

示例输出如下。

```
{
    "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-*1234567890abcdef01234567890abcde*
- 审计 (**audit**) – kube-apiserver-audit-*1234567890abcdef01234567890abcde*
- 身份验证器 (**authenticator**) – authenticator-*1234567890abcdef01234567890abcde*
- 控制器管理器 (**controllerManager**) – kube-controller-manager-*1234567890abcdef01234567890abcde*
- 调度器 (**scheduler**) – kube-scheduler-*1234567890abcdef01234567890abcde*

4. 查看日志流的事件。

   例如，在查看 kube-apiserver-*1234567890abcdef01234567890abcde* 顶部时，您应看到集群的初始 API 服务器标志。

> **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。

使用 AWS CloudTrail 记录 Amazon EKS API 调用

Amazon ECR 与 AWS CloudTrail 集成。CloudTrail 服务在 Amazon EKS 中提供用户、角色或 AWS 服务的操作记录。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，则仍可在 CloudTrail 控制台中的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《AWS CloudTrail 用户指南》。

主题

CloudTrail 中的 Amazon EKS 信息

在您创建 AWS 账户时，还将在您的 AWS 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 AWS 服务事件一同保存在 Event history（事件历史记录）中。您可以在 AWS 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AWS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AWS 区域。此跟踪记录来自 AWS 区域 分区中所有 AWS 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 AWS 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包含有关发出请求的 IAM 身份之类型的详细信息，以及使用了哪些凭证。如果使用的是临时凭证，则该条目显示凭证是如何获取的。

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AWS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。

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"
}
```

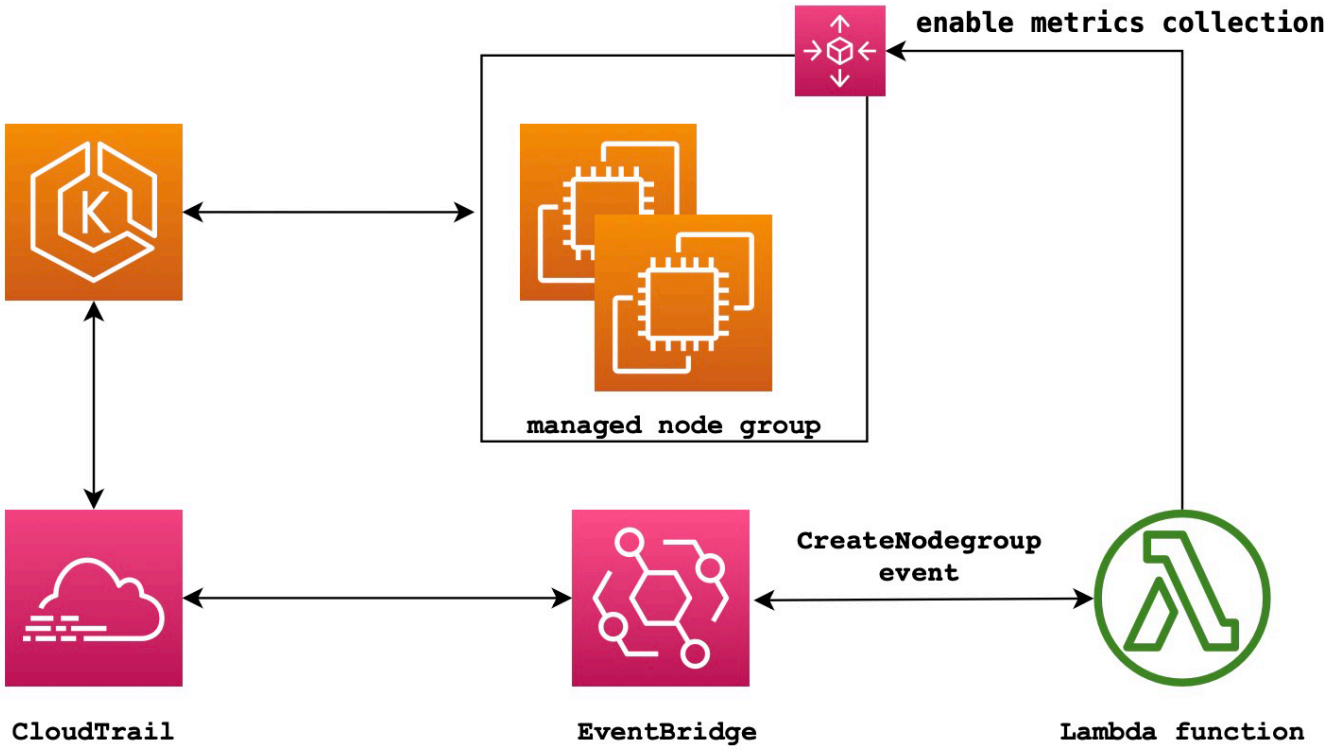
启用自动扩缩组指标收集

本主题介绍如何使用 AWS Lambda 和 AWS CloudTrail 启用自动扩缩组指标收集。Amazon EKS 不会自动为托管节点创建的自动扩缩组启用群组指标收集。

您可以使用自动扩缩组指标来跟踪自动扩缩组中的更改并设置阈值警报。自动扩缩组指标在 Auto Scaling 控制台或 Amazon CloudWatch 控制台中提供。启用后，自动扩缩组每分钟向 Amazon CloudWatch 发送采样数据。启用这些指标无需支付费用。

通过启用自动扩缩组指标收集，您将能够监控托管节点组的扩缩情况。自动扩缩组指标可报告自动扩缩组的最小、最大和所需大小。如果节点组中的节点数量小于最小大小（这表示节点组运行状况不佳），您可以创建警报。跟踪节点组大小在调整最大数量时也很有用，以便您的数据面板不会耗尽容量。

当您创建托管节点组时，AWS CloudTrail 会向 Amazon EventBridge 发送一个 `CreateNodegroup` 事件。通过创建与 `CreateNodegroup` 事件匹配的 Amazon EventBridge 规则，您可以触发 Lambda 函数，来为与托管节点组关联的自动扩缩组启用群组指标收集。



启用自动扩缩组指标收集

1. 为 Lambda 创建 IAM 角色。

```
LAMBDA_ROLE=$(aws iam create-role \
  --role-name lambda-asg-enable-metrics \
  --assume-role-policy-document '{"Version": "2012-10-17","Statement":
 [{ "Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action":
 "sts:AssumeRole"}]}' \
  --output text \
  --query 'Role.Arn')
echo $LAMBDA_ROLE
```

2. 创建允许描述 Amazon EKS 节点组和启用自动扩缩组指标收集的策略。

```
cat > /tmp/lambda-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
            "eks:DescribeNodegroup",
            "autoscaling:EnableMetricsCollection"
            ],
        "Resource": [
            "*"
        ]
      }
    ]
}
EOF
LAMBDA_POLICY_ARN=$(aws iam create-policy \
  --policy-name lambda-asg-enable-metrics-policy \
  --policy-document file:///tmp/lambda-policy.json \
  --output text \
  --query 'Policy.Arn')
echo $LAMBDA_POLICY_ARN
```

3. 向 Lambda 的 IAM 角色附加策略。

```
aws iam attach-role-policy \
  --policy-arn $LAMBDA_POLICY_ARN \
  --role-name lambda-asg-enable-metrics
```

4. 添加 AWSLambdaBasicExecutionRole 托管策略，该策略具有函数将日志写入 CloudWatch Logs 所需的权限。

```
aws iam attach-role-policy \
  --role-name lambda-asg-enable-metrics \
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

5. 创建 Lambda 代码。

```
cat > /tmp/lambda-handler.py <<EOF
import json
```

```
import boto3
import time
import logging

eks = boto3.client('eks')
autoscaling = boto3.client('autoscaling')

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    ASG_METRICS_COLLLECTION_TAG_NAME = "ASG_METRICS_COLLLECTION_ENABLED"
    initial_retry_delay = 10
    attempts = 0

    #print(event)

    if not event["detail"]["eventName"] == "CreateNodegroup":
        print("invalid event.")
        return -1

    clusterName = event["detail"]["requestParameters"]["name"]
    nodegroupName = event["detail"]["requestParameters"]["nodegroupName"]
    try:
        metricsCollectionEnabled = event["detail"]["requestParameters"]["tags"]
[ASG_METRICS_COLLLECTION_TAG_NAME]
    except KeyError:
        print(ASG_METRICS_COLLLECTION_TAG_NAME, "tag not found.")
        return

    # Check if metrics collection is enabled in tags
    if metricsCollectionEnabled.lower() != "true":
        print("Metrics collection is not enabled in nodegroup tags.")
        return

    # Get the name of the associated autoscaling group
    print("Getting the autoscaling group name for nodegroup=", nodegroupName, ",
 cluster=", clusterName )
    for i in range(0,10):
        try:
            autoScalingGroup =
 eks.describe_nodegroup(clusterName=clusterName,nodegroupName=nodegroupName)
["nodegroup"]["resources"]["autoScalingGroups"][0]["name"]
        except:
```

```
            attempts += 1
            print("Failed to obtain the associated autoscaling group for
 nodegroup", nodegroupName, "Retrying in", initial_retry_delay*attempts,
 "seconds.")
            time.sleep(initial_retry_delay*attempts)
        else:
            break

    print("Enabling metrics collection on autoscaling group ", autoScalingGroup)

    # Enable metrics collection in the autoscaling group
    try:
        enableMetricsCollection =
 autoscaling.enable_metrics_collection(AutoScalingGroupName=autoScalingGroup,Granularity="1
    except:
        print("Unable to enable metrics collection on nodegroup=",nodegroup)
    print("Enabled metrics collection on nodegroup", nodegroupName)
EOF
```

6. 创建部署程序包。

```
cd /tmp
zip function.zip lambda-handler.py
```

7. 创建一个 Lambda 函数。

```
LAMBDA_ARN=$(aws lambda create-function --function-name asg-enable-metrics-
collection \
  --zip-file fileb://function.zip --handler lambda-handler.lambda_handler \
  --runtime python3.9 \
  --timeout 600 \
  --role $LAMBDA_ROLE \
  --output text \
  --query 'FunctionArn')
echo $LAMBDA_ARN
```

8. 创建 EventBridge 规则。

```
RULE_ARN=$(aws events put-rule --name CreateNodegroupRuleToLambda \
  --event-pattern "{\"source\":[\"aws.eks\"],\"detail-type\":[\"AWS API Call via
 CloudTrail\"],\"detail\":{\"eventName\":[\"CreateNodegroup\"],\"eventSource\":
[\"eks.amazonaws.com\"]}}" \
  --output text \
```

```
  --query 'RuleArn')
echo $RULE_ARN
```

9. 添加 Lambda 函数作为目标。

```
aws events put-targets --rule CreateNodegroupRuleToLambda \
 --targets "Id"="1","Arn"="$LAMBDA_ARN"
```

10. 添加允许 EventBridge 调用 Lambda 函数的策略。

```
aws lambda add-permission \
  --function-name asg-enable-metrics-collection \
  --statement-id CreateNodegroupRuleToLambda \
  --action 'lambda:InvokeFunction' \
  --principal events.amazonaws.com \
  --source-arn $RULE_ARN
```

对于您使用 ASG_METRICS_COLLLECTION_ENABLED 设置为 TRUE 进行标记的任何托管节点组，Lambda 函数可为这些节点组启用自动扩缩组指标收集。要确认已启用 Auto Scaling group metrics collection（自动扩缩组指标收集），请在 Amazon EC2 控制台中导航到关联的自动扩缩组。在 Monitoring（监控）选项卡中，您可看到已激活 Enable（启用）复选框。

对 ADOT Operator 的 Amazon EKS 附加组件支持

Amazon EKS 支持使用 AWS Management Console、AWS CLI 和 Amazon EKS API 来安装和管理 AWS Distro for OpenTelemetry (ADOT) Operator。这样，在 Amazon EKS 上运行的应用程序就能轻松地向多个监控服务选项发送指标和跟踪数据，比如 Amazon CloudWatch、Prometheus 和 X-Ray。

有关更多信息，请参阅适用于 OpenTelemetry 的 AWS Distro 文档中的使用 EKS 插件且适用于 OpenTelemetry 的 AWS Distro 入门。

与 Amazon EKS 集成的其他 AWS 服务

除了其他部分介绍的服务外，Amazon EKS 还与其他 AWS 服务配合使用，以提供其他解决方案。本主题介绍了使用 Amazon EKS 添加功能的一些其他服务，或 Amazon EKS 用来执行任务的服务。

主题

利用 AWS CloudFormation 创建 Amazon EKS 资源

Amazon EKS 与 AWS CloudFormation 集成，后者是一项服务，可帮助您对AWS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AWS资源（如 Amazon EKS 集群）的模板，而 AWS CloudFormation 将负责为您设置和配置这些资源。

在您使用 AWS 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AWS账户和区域中反复配置相同的资源。

Amazon EKS 和 AWS CloudFormation 模板

要为 Amazon EKS 和相关服务预置和配置资源，您必须了解 AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 AWS CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 AWS CloudFormation Designer 的帮助下开始使用 AWS CloudFormation 模板。有关更多信息，请参阅 AWS CloudFormation 用户指南中的什么是 AWS CloudFormation Designer？。

Amazon EKS 支持在 AWS 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 AWS CloudFormation 用户指南中的 Amazon EKS 资源类型参考。

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation 命令行界面用户指南

Amazon EKS 和 AWS 本地区域

AWS Local Zone 是在地理上靠近用户的 AWS 区域的扩展。Local Zones 有自己的 Internet 连接并支持 AWS 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、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、AWS 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。

网络架构

- Amazon EKS 托管的 Kubernetes 控制面板始终在 AWS 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 AWS 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。
- 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

AWS 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。深度学习容器提供具有 TensorFlow、NVIDIA CUDA（适用于 GPU 实例）和 Intel MKL（适用于 CPU 实例）库的优化环境，并且在 Amazon ECR 中可用。

要开始使用 Amazon EKS 上的 AWS 深度学习容器，请参阅《AWS 深度学习容器开发者指南》中的 Amazon EKS 设置。

Amazon VPC Lattice

Amazon VPC Lattice 是一项完全托管的应用程序网络服务，它直接内置在 AWS 网络基础设施中，您可以将其用于连接、保护和监控您在多个账户和虚拟私有云（VPC）中的服务。使用 Amazon EKS，您可以通过使用 AWS Gateway API 控制器（这是 Kubernetes Gateway API 的一种实施）来利用 Amazon VPC Lattice。使用 Amazon VPC Lattice，您可以以简单一致的方式设置具有标准 Kubernetes 语义的跨集群连接。要开始将 Amazon VPC Lattice 与 Amazon EKS 结合使用，请参阅 AWS Gateway API 控制器用户指南。

AWS Resilience Hub

AWS Resilience Hub 通过分析 Amazon EKS 集群的基础设施来评估其弹性。AWS Resilience Hub 使用 Kubernetes 基于角色的访问控制（RBAC）配置来评测部署到集群的 Kubernetes 工作负载。有关更多信息，请参阅《AWS Resilience Hub 用户指南》中的为 Amazon EKS 集群启用 AWS Resilience Hub 访问。

使用 Amazon GuardDuty 检测威胁

Amazon GuardDuty 是一项威胁检测服务，有助于保护您的账户、容器、工作负载和 AWS 环境中的数据。GuardDuty 使用机器学习（ML）模型以及异常和威胁检测功能，持续监控不同的日志源和运行时活动，以识别环境中的潜在安全风险和恶意活动并确定其优先级。

除其他功能外，GuardDuty 还提供以下两项功能，用于检测 EKS 集群面临的潜在威胁：EKS 保护和运行时监控。

EKS 保护

> 此功能提供威胁检测覆盖范围，帮助您通过监控关联的 Kubernetes 审计日志来保护 Amazon EKS 集群。Kubernetes 审计日志可捕获集群内的连续操作，包括来自用户、使用 Kubernetes API 的应

用程序以及控制面板的活动。例如，GuardDuty 可以识别出未经身份验证的用户执行的 API 调用，这些调用可能用于篡改 Kubernetes 集群中的资源。

启用 EKS 保护后，GuardDuty 将只能访问 Amazon EKS 审计日志，从而持续进行威胁检测。如果 GuardDuty 发现集群存在潜在威胁，就会生成具有特定类型的关联的 Kubernetes 审计日志调查发现。有关 Kubernetes 审计日志中可用的调查发现类型的更多信息，请参阅《Amazon GuardDuty User Guide》中的 Kubernetes audit logs finding types。

有关更多信息，请参阅《Amazon GuardDuty User Guide》中的 EKS Protection。

运行时监控

此功能可监控和分析操作系统级事件、网络事件和文件事件，帮助您检测环境中特定 AWS 工作负载中的潜在威胁。

启用运行时监控并在 Amazon EKS 集群中安装 GuardDuty 代理后，GuardDuty 就会开始监控与此集群关联的运行时事件。如果 GuardDuty 发现集群存在潜在威胁，就会生成关联的运行时监控调查发现。例如，威胁可能会从破坏单个容器开始，而这种容器通常在运行易受攻击的 Web 应用程序。此 Web 应用程序可能拥有对底层容器和工作负载的访问权限。在这种情况下，错误配置的凭证可能会导致对账户及其所存储数据的访问权限扩大。

要配置运行时监控，可将 GuardDuty 代理作为 Amazon EKS 附加组件安装到集群中。有关附加组件的更多信息，请参阅 Amazon EKS 提供的可用 Amazon EKS 附加组件。

有关更多信息，请参阅《Amazon GuardDuty User Guide》中的 Runtime Monitoring。

将 Amazon Security Lake 与 Amazon EKS 结合使用

Amazon Security Lake 是一项完全托管的安全数据湖服务，通过它，您可以集中来自各种来源（包括 Amazon EKS）的安全数据。通过将 Amazon EKS 与 Security Lake 集成，您可以更深入地了解在您的 Kubernetes 资源上执行的活动，并增强 Amazon EKS 集群的安全状况。

> **Note**
>
> 有关将 Security Lake 与 Amazon EKS 结合使用和设置数据来源的更多信息，请参阅 Amazon Security Lake 文档。

将 Security Lake 与 Amazon EKS 结合使用的益处

集中安全数据 – Security Lake 自动收集和集中来自 Amazon EKS 集群的安全数据，以及来自其他 AWS 服务、SaaS 提供商、本地来源和第三方来源的数据。这将提供整个组织的安全状况的全面视图。

标准化数据格式 – Security Lake 将收集的数据转换为标准开源架构 – 开放网络安全架构框架（OCSF）格式。这种标准化使分析和集成其他安全工具和服务变得更加容易。

改进了威胁检测 – 通过分析集中式安全数据，包括 Amazon EKS 控制面板日志，您可以更有效地检测 Amazon EKS 集群中可能存在的可疑活动。这有助于及时识别和响应安全事件。

简化数据管理 – Security Lake 通过可自定义的保留设置和复制设置来管理安全数据的生命周期。这简化了数据管理任务，并确保您可以保留用于合规和审计目的所需的数据。

为 Amazon EKS 启用 Security Lake

要开始将 Security Lake 与 Amazon EKS 结合使用，请执行以下步骤：

1. 为 EKS 集群启用 Amazon EKS 控制面板日志记录。有关详细说明，请参阅启用和禁用控制面板日志。
2. 将 Amazon EKS 审核日志作为来源添加到 Security Lake 中。然后，Security Lake 将开始收集在您的 EKS 集群中运行的 Kubernetes 资源上执行的活动的相关深入信息。
3. 根据您的要求在 Security Lake 中为您的安全数据配置保留和复制设置。
4. 使用存储在 Security Lake 中的标准化 OCSF 数据进行事件响应、安全分析以及与其他 AWS 服务或第三方工具的集成。例如，您可以使用 Amazon OpenSearch Ingestion 从 Amazon Security Lake 数据中生成安全见解。

在 Security Lake 中分析 EKS 日志

Security Lake 将 EKS 日志事件标准化为 OCSF 格式，从而使分析数据并将其与其他安全事件关联起来更加容易。您可以使用各种工具和服务，例如 Amazon Athena、Amazon QuickSight 或第三方安全分析工具，来查询和可视化标准化数据。

有关 EKS 日志事件的 OCSF 映射的详细信息，请参阅 OCSF GitHub 存储库中的映射参考。

Amazon Detective

Amazon Detective 可帮助您分析、调查和快速识别安全结果或可疑活动的根本原因。Detective 会自动从AWS资源收集日志数据。然后，它使用机器学习、统计分析和图形理论生成可视化效果，帮助更快、更高效地进行安全调查。Detective 的预构建数据聚合、摘要和上下文可有助于分析和确定潜在安全问题的性质和程度。有关更多信息，请参阅 Amazon Detective 用户指南。

Detective 将 Kubernetes 和 AWS 数据组织到发现结果中，例如：

- Amazon EKS 集群详细信息，包括创建集群的 IAM 身份和集群的服务角色。可以使用 Detective 调查这些 IAM 身份的 AWS 和 Kubernetes API 活动。
- 容器详细信息，例如映像和安全上下文。此外，您还可以查看已终止 Pods 的详细信息。
- Kubernetes API 活动，包括 API 活动的总体趋势和特定 API 调用的详细信息。例如，您可以显示在选定时间范围内发出的成功和失败 Kubernetes API 调用的数量。此外，有关新观察到的 API 调用的部分可能有助于识别可疑活动。

Amazon EKS 审核日志是一个可选的数据来源软件包，可以添加到您的 Detective 行为图中。您可以查看自己的账户中可用的可选来源软件包及其状态。有关更多信息，请参阅 Amazon Detective 用户指南中的适用于 Detective 的 Amazon EKS 审核日志。

将 Amazon Detecty 与 Amazon EKS 结合使用

查看 Amazon EKS 集群的发现结果

在查看发现结果之前，必须于集群所在的同一 AWS 区域 内启用 Detective 至少 48 小时。有关更多信息，请参阅 Amazon Detective 用户指南中的设置 Amazon Detective。

1. 打开 Detective 控制台，网址为 https://console.aws.amazon.com/detective/。
2. 从左侧导航窗格中选择搜索。
3. 选择选择类型，然后选择 EKS 集群。
4. 输入集群名称或 ARN，然后选择搜索。
5. 在搜索结果中，选择要查看其活动的集群的名称。有关您可以查看的内容的更多信息，请参阅 Amazon Detective 用户指南中的涉及 Amazon EKS 集群的整体 Kubernetes API 活动。

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。如果您需要对特定的 Amazon EKS 区域进行问题排查，请参阅单独的 IAM 故障排除、排查 Amazon EKS Connector 中的问题，以及使用 EKS 插件对 ADOT 进行问题排查主题。

有关其他故障排除信息，请参阅 AWS re:Post 上的有关 Amazon Elastic Kubernetes Service 的知识中心内容。

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。

有些可用区是集群无法驻留的。将您的子网所在的可用区与 子网要求和注意事项 中的可用区列表进行比较。

节点未能加入集群

有几种常见原因会阻止节点加入集群：

- 如果节点是托管节点，Amazon EKS 会在您创建节点组时向 `aws-auth ConfigMap` 中添加条目。如果该条目已被移除或修改，则需要重新添加该条目。要了解更多信息，请在您的终端中输入 **`eksctl create iamidentitymapping --help`**。您可以通过将以下命令中的 *`my-cluster`* 替换为集群名称，然后运行修改后的命令来查看当前的 `aws-auth ConfigMap` 条目：**`eksctl get iamidentitymapping --cluster my-cluster`**。您指定的角色的 ARN 不能包含 `/` 之外的路径。例如，如果您的角色名称为 `development/apps/my-role`，则需要在为该角色指定 ARN 时将其更改为 `my-role`。确保指定了节点 IAM 角色 ARN（非实例配置文件 ARN）。

  如果节点是自行管理的，并且您尚未为该节点的 IAM 角色的 ARN 创建访问条目，请运行与托管节点列出的相同命令。如果您为节点 IAM 角色的 ARN 创建了访问条目，则可能无法在访问条目中正确配

置该条目。确保将节点 IAM 角色 ARN（非实例配置文件 ARN）指定为 aws-auth ConfigMap 条目或访问条目中的主体 ARN。有关访问条目的更多信息，请参阅 管理访问条目。

- 节点 AWS 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。

键	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 AWS 区域 的 AWS STS 端点。要启用该区域，请参阅在 AWS 区域中激活和停用 AWS STS。
- 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。
- 如果托管节点组中的节点未在 15 分钟内连接到集群，则将发出运行状况问题“nodecreationFailure”，控制台状态将设置为 Create failed。对于启动时间较慢的 Windows AMI，可以使用快速启动来解决此问题。

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 AWS 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，则说明您未针对 Amazon EKS 正确配置 kubectl，或您使用的 IAM 主体（角色或用户）的凭证未映射到对 Amazon EKS 集群上的 Kubernetes 对象具有足够权限的 Kubernetes 用户名。

- `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"`

这可能是由以下原因之一导致的：

- 集群是使用一个 IAM 主体的凭证创建的，并且 kubectl 配置为使用另一个 IAM 主体的凭证。要解决此问题，请更新您的 kube config 文件以使用创建集群的凭证。有关更多信息，请参阅 为 Amazon EKS 集群创建或更新 kubeconfig 文件。
- 如果您的集群满足 管理访问条目 的“先决条件”部分中的最低平台要求，则您的 IAM 主体不存在访问条目。如果存在，则说明没有为其定义必要的 Kubernetes 组名称，或者没有与之关联的正确访问策略。有关更多信息，请参阅 管理访问条目。
- 如果您的集群不符合 管理访问条目 中的最低平台要求，则 aws-auth ConfigMap 中不存在具有您的 IAM 主体的条目。如果存在，则不会映射到 Kubernetes 组名称，这些名称已绑定到具有必要权限的 Kubernetes Role 或 ClusterRole。有关 Kubernetes 基于角色授权（RBAC）对象的更多信息，请参阅 Kubernetes 文档中的使用 RBAC 鉴权。您可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令来查看当前的 aws-auth ConfigMap 条目：**eksctl get iamidentitymapping --cluster *my-cluster***。如果具有您的 IAM 主体的 ARN 的条目不在 ConfigMap 中，请在您的终端中输入 **eksctl create iamidentitymapping --help** 以了解如何创建一个。

如果安装和配置 AWS CLI，则可配置您使用的 IAM 凭证。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的配置 AWS CLI。如果您分派 IAM 角色访问集群上的 Kubernetes 对象，则也可以配置 kubectl 以使用 IAM 角色。有关更多信息，请参阅 为 Amazon EKS 集群创建或更新 kubeconfig 文件。

hostname doesn't match

系统的 Python 版本必须为 2.7.9 或更高版本。否则，在 AWS 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 Requests Frequently Asked Questions》中的 What are "hostname doesn't match" errors?

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
```

Instances failed to join the Kubernetes cluster

如果您在AWS Management Console中收到 Instances failed to join the Kubernetes cluster 错误，请确保已启用集群的私有端点访问，或者您已正确配置 CIDR 块以用于公有端点访问。有关更多信息，请参阅 Amazon EKS 集群端点访问控制。

托管节点组错误代码

如果您的托管节点组遇到硬件运行状况问题，则 Amazon EKS 返回错误代码以帮助您诊断问题。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。下面的列表介绍了错误代码。

AccessDenied

Amazon EKS 或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。有关解决常见原因的更多信息，请参阅 修复托管节点组的 AccessDenied 错误的常见原因。私有 Windows AMI 也可能导致出现此错误代码以及 Not authorized for images 错误消息。有关更多信息，请参阅 Not authorized for images。

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自动扩缩组。您可以重新创建具有相同设置的自动扩缩组进行恢复。

ClusterUnreachable

Amazon EKS 或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

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

自动扩缩组在尝试启动实例时出现故障。

NodeCreationFailure

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，请参阅子网要求和注意事项 和Amazon EKS 安全组要求和注意事项。
- 节点和 VPC 必须满足 私有集群要求 中的要求。

InstanceLimitExceeded

AWS 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** 错误的常见原因

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。

Not authorized for images

Not authorized for images 错误消息的一个潜在原因是使用私有 Amazon EKS Windows AMI 启动 Windows 托管节点组。发布新的 Windows AMI 后，AWS 会将已超过 4 个月的 AMI 设为私有，这会使得这些 AMI 不再可访问。如果您的托管节点组使用的是私有 Windows AMI，则请考虑更新您的 Windows 托管节点组。虽然我们不能保证可以提供对已设为私有的 AMI 的访问权限，但您可以通过向 AWS Support 提交服务单来请求访问权限。有关更多信息，请参阅 Amazon EC2 用户指南中的补丁、安全更新和 AMI ID。

节点处于 NotReady 状态

如果您的节点进入 NotReady 状态，则可能表示该节点运行状况不佳，无法计划新的 Pods。这可能是由于各种原因造成的，例如节点缺少足够的 CPU、内存或可用磁盘空间资源。

对于 Amazon EKS 优化的 Windows AMI，在 kubelet 配置中没有默认指定的计算资源预留。为帮助防止出现资源问题，您可以为系统进程预留计算资源，方法是为 kubelet 提供 kube-reserved 和/或 system-reserved 的配置值。您可以使用引导脚本中的 -KubeletExtraArgs 命令行参数来执行此操作。有关更多信息，请参阅 Kubernetes 文档中的为系统进程守护程序预留计算资源和本用户指南中的引导脚本配置参数。

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
```

这可能是由以下原因之一导致的：

1. 您的集群上要么没有 aws-auth ConfigMap，要么其中不包含您为节点配置的 IAM 角色的条目。

   如果节点满足以下条件之一，则此 ConfigMap 条目是必需的：

   - 集群中任何 Kubernetes 或平台版本的托管节点。
   - 集群中的自行管理的节点，该节点早于 管理访问条目 主题先决条件部分列出的平台版本之一。

   要解决此问题，可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令 **eksctl get iamidentitymapping --cluster *my-cluster*** 来查看 ConfigMap 中的现有条目。如果您收到来自该命令的错误消息，则可能是因为您的集群没有 aws-auth ConfigMap。以下命令将条目添加到 ConfigMap。如果 ConfigMap 不存在，该命令也会创建一个。将 *111122223333* 替换为 IAM 角色的 AWS 账户 ID，将 *myAmazonEKSNodeRole* 替换为节点角色的名称。

   ```
   eksctl create iamidentitymapping --cluster my-cluster \
       --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
    system:bootstrappers,system:nodes \
       --username system:node:{{EC2PrivateDNSName}}
   ```

   您指定的角色的 ARN 不能包含 / 之外的路径。例如，如果您的角色名称为 development/apps/my-role，则需要在指定角色的 ARN 时将其更改为 my-role。确保指定了节点 IAM 角色 ARN（非实例配置文件 ARN）。

2. 您的自行管理的节点位于集群中，其平台版本为 管理访问条目 主题先决条件中列出的最低版本，但该节点 IAM 角色的 aws-auth ConfigMap 中未列出条目（参见上一项），或者该角色不存在访问条目。要解决此问题，可以通过将以下命令中的 *my-cluster* 替换为集群名称，然后运行修改后的命令 **aws eks list-access-entries --cluster-name *my-cluster*** 来查看现有访问条目。以下命令为节点的 IAM 角色添加访问条目。将 *111122223333* 替换为 IAM 角色的 AWS 账户 ID，将 *myAmazonEKSNodeRole* 替换为节点角色的名称。如果您有 Windows 节点，请将 *EC2_Linux* 替换为 **EC2_Windows**。确保指定了节点 IAM 角色 ARN（非实例配置文件 ARN）。

   ```
   aws eks create-access-entry --cluster-name my-cluster --principal-arn
    arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_Linux
   ```

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 AWS 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 AWS 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 更新 VPC 准入 Webhook 证书。如果您的集群和平台版本高于 Windows 支持先决条件部分中列出的版本，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。

在升级控制面板前，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升级为新的 Kubernetes 版本。Amazon EKS `update-cluster-version` API 会拒绝请求，直到您将所有 Amazon EKS 托管节点升级为当前集群版本。Amazon EKS 提供 API 来升级托管节点。有关升级托管节点组 Kubernetes 版本的信息，请参阅 更新托管节点组。要升级 Fargate 节点的版本，请删除节点所表示的 pod，然后在升级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。

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` 参数正在传递给节点的引导脚本。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 Linux/Bottlerocket AMI 一起提供的 `bootstrap.sh` 文件。

对于 Kubernetes API 服务器请求的 HTTP 401 未经授权错误响应

如果集群上的 Pod 的服务账户令牌已过期，您将会看到这些错误。

Amazon EKS 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。

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
```

示例输出如下。

```
eksClusterRole
```

解决方案

创建具有相同名称的新集群 IAM 角色。

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
```

示例输出如下。

```
[
"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"
```

示例输出如下。

```
[
"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，则需要更改集群使用的子网。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。

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
```

示例输出如下。

```
[
    "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"
```

示例输出如下。

```
[
"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，则需要更改集群使用的安全组。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。

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，或者需要更改集群使用的子网，以使用具有足够可用 IP 地址的子网。
- 您在创建集群时启用了密钥加密，但您指定的 AWS 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

集群运行状况常见问题解答和错误代码以及解析路径

Amazon EKS 会检测您的 EKS 集群和集群基础设施存在的问题，并将其存储在集群运行状况中。借助集群运行状况信息，您可以更快地检测、排查并解决集群问题。这使您能够创建更安全、更新的应用程序环境。此外，由于必要的基础设施或集群配置存在问题，您可能无法升级到 Kubernetes 的较新版本或 Amazon EKS 无法在已降级的集群上安装安全更新。Amazon EKS 可能需要 3 小时来检测问题或检测问题是否已解决。

Amazon EKS 集群的运行状况由 Amazon EKS 及其用户共同负责。您负责管理 IAM 角色和 Amazon VPC 子网的必备基础设施，以及必须提前提供的其他必要基础设施。Amazon EKS 检测此基础设施和集群的配置发生的变化。

要在 Amazon EKS 控制台中访问您的集群运行状况，请在 Amazon EKS 集群详细信息页面的概述选项卡中查找名为运行状况问题的部分。也可以通过在 EKS API 中调用 `DescribeCluster` 操作来获得这些数据，例如从 AWS Command Line Interface 内部调用。

为什么应该使用此功能？

> 您可以更清楚地了解 Amazon EKS 集群的运行状况，快速诊断和修复任何问题，而无需花时间调试或提出 AWS 支持案例。例如：您不小心删除了 Amazon EKS 集群的子网，Amazon EKS 将无法创建跨账户网络接口和 Kubernetes AWS CLI 命令，例如 `kubectl exec` 或 `kubectl` 日志。这些将失败并显示错误：`Error from server: error dialing backend: remote error: tls: internal error.` 现在您将看到一个 Amazon EKS 运行状况问题，错误消息为：`subnet-da60e280 was deleted: could not create network interface`。

此功能如何与其他 AWS 服务相关或结合使用？

> IAM 角色和 Amazon VPC 子网是集群运行状况检测到问题的两个必备基础设施示例。如果这些资源配置不正确，则此功能将返回详细信息。

存在运行状况问题的集群是否会产生费用？

> 是的，每个 Amazon EKS 集群都按标准的 Amazon EKS 定价计费。集群运行状况功能不收取额外费用。

此功能是否可以与 AWS Outposts 上的 Amazon EKS 集群结合使用？

> 是的，已检测到 AWS Cloud 中的 EKS 集群存在集群问题，包括 AWS Outposts 上的扩展集群和 AWS Outposts 上的本地集群。集群运行状况无法检测到 Amazon EKS Anywhere 或 Amazon EKS Distro（EKS-D）存在的问题。

当检测到新问题时，我能否收到通知？

不，您需要查看 Amazon EKS 控制台或调用 EKS `DescribeCluster` API。

控制台是否会就运行状况问题向我发出警告？

是的，任何存在运行状况问题的集群都将在控制台顶部显示一个横幅。

前两列是 API 响应值所需的列。Health ClusterIssue 对象的第三个字段是 resourceIds，其返回值取决于问题类型。

代码	消息	ResourceIds	集群是否可恢复？	
SUBNET_NOT_FOUND	我们找不到当前与您的集群关联的一个或多个子网。调用 Amazon EKS update-cluster-config API 更新子网。	子网 ID	是	
SECURITY_GROUP_NOT_FOUND	我们找不到当前与您的集群关联的一个或多个安全组。调用 Amazon EKS update-cluster-config API 来更新安全组	安全组 ID	是	
IP_NOT_AVAILABLE	与集群关联的一个或多个子网没有足够的 IP 地址供 Amazon EKS 执行集群管理操作。使用 Amazon EKS update-cluster-config API 释放子网中的地址，或者将不同的子网关联到您的集群。	子网 ID	是	
VPC_NOT_FOUND	我们找不到与您的集群关联的 VPC。您必须删除并重新创建集群。	VPC ID	否	
ASSUME_ROLE_ACCESS_DENIED	您的集群未使用 Amazon EKS service-linked-role。我们无法分派与您的集群关联的角色来执行所需的 Amazon EKS 管理操作。检查该	集群 IAM 角色	是	

代码	消息	ResourceIds	集群是否可恢复？	
	角色是否存在并且是否具有所需的信任策略。			
PERMISSION_ACCESS_DENIED	您的集群未使用 Amazon EKS service-linked-role。与您的集群关联的角色未向 Amazon EKS 授予足够的权限来执行所需的管理操作。检查附加到集群角色的策略，以及是否应用了任何单独的拒绝策略。	集群 IAM 角色	是	
ASSUME_ROLE_ACCESS_DENIED_USING_SLR	我们无法分派 Amazon EKS 集群管理 service-linked-role。检查该角色是否存在并且是否具有所需的信任策略。	Amazon EKS service-linked-role	是	
PERMISSION_ACCESS_DENIED_USING_SLR	Amazon EKS 集群管理 service-linked-role 未向 Amazon EKS 授予足够的权限来执行所需的管理操作。检查附加到集群角色的策略，以及是否应用了任何单独的拒绝策略。	Amazon EKS service-linked-role	是	
OPT_IN_REQUIRED	您的账户没有订阅 Amazon EC2 服务。在您的账户设置页面更新您的账户订阅。	不适用	是	
STS_REGIONAL_ENDPOINT_DISABLED	STS 区域端点已禁用。启用 Amazon EKS 的端点以执行所需的集群管理操作。	不适用	是	
KMS_KEY_DISABLED	与您的集群关联的 AWS KMS 密钥已禁用。重新启用密钥以恢复集群。	这些区域有：KMS Key Arn	是	

代码	消息	ResourceIds	集群是否可恢复？	
KMS_KEY_NOT_FOUND	我们找不到与您的集群关联的 AWS KMS 密钥。您必须删除并重新创建集群。	这些区域有：KMS Key ARN	否	
KMS_GRANT_REVOKED	已撤销对与您的集群关联的 AWS KMS 密钥的授权。您必须删除并重新创建集群。	这些区域有：KMS Key Arn	否	

Amazon EKS Connector

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 AWS，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 需要一个代理，该代理是 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

Amazon EKS Connector 能够将以下类型的 Kubernetes 集群连接到 Amazon EKS。

- 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《AWS 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：

- 首次注册集群时，您需要创建 Amazon EKS Connector 服务相关角色。
- 您必须创建 Amazon EKS Connector 代理 IAM 角色。有关详细信息，请参阅Amazon EKS Connector IAM 角色。

要为 IAM 主体启用集群和工作负载视图权限，请将 `eks-connector` 和 Amazon EKS Connector 集群角色应用到集群。按 向 IAM 主体授予查看集群上的 Kubernetes 资源的访问权限 中的步骤操作。

连接外部集群

您可以在以下过程中使用多种方法将外部 Kubernetes 集群连接到 Amazon EKS。此过程包括两个步骤：向 Amazon EKS 注册集群，在集群中安装 `eks-connector` 代理。

> **Important**
>
> 您必须在完成第一步后 3 天内（注册到期之前）完成第二步。

Connector 方法

在用过注册集群的每种方法之后，并不是所有安装代理的方法都可以使用。下表列出了每种注册方法以及可以使用的安装代理的方法。

步骤	方法		
注册集群	AWS Management Console	AWS Command Line Interface	`eksctl`
安装 代理	Helm，YAML 清单	Helm，YAML 清单	YAML 清单

先决条件

- 确保已创建 Amazon EKS Connector 代理角色。按照 创建 Amazon EKS Connector 代理角色 中的步骤操作。
- 您必须拥有以下权限才能注册集群：
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

步骤 1：注册集群

AWS CLI

先决条件

- 必须安装 AWS CLI。要进行安装或升级，请参阅安装 AWS CLI。

要使用 AWS CLI 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。

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

示例输出如下。

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
```

```
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

在下一步中使用 `aws-region`、`activationId` 和 `activationCode` 值。

AWS Management Console

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。如果您想使用 Helm 图表，请复制 `helm install` 命令并继续下一步。如果要使用 YAML 清单，请选择下载 YAML 文件，将清单文件下载到本地驱动程序。

> **Important**
>
> - 这是您复制 `helm install` 命令或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。
> - 对于注册的集群，命令或清单文件只能使用一次。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。

eksctl

先决条件

- 必须安装 `eksctl` 版本 `0.68` 或更高版本。要安装或对其升级，请参阅 开始使用 Amazon EKS – eksctl。

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
```

输出示例：

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-
connector-console-dashboard-full-access-group.yaml" give full EKS Console access
 to IAM identity "<aws-arn>", edit if required; read https://eksctl.io/usage/
eks-connector for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

这将在本地计算机上创建文件。必须在 3 天内将这些文件应用到外部集群，否则注册将会过期。

2. 在可以访问集群的终端中，应用 `eks-connector-binding.yaml` 文件：

```
kubectl apply -f eks-connector-binding.yaml
```

步骤 2：安装 eks-connector 代理

Helm chart

1. 如果在上一步中使用了 AWS CLI，请将以下命令中的 ACTIVATION_CODE 和 ACTIVATION_ID 分别替换为 activationId、和 activationCode 值。将 aws-region 替换为在上一步中使用的 AWS 区域。然后运行命令在注册集群上安装 eks-connector 代理：

   ```
   $ helm install eks-connector \
     --namespace eks-connector \
     oci://public.ecr.aws/eks-connector/eks-connector-chart \
     --set eks.activationCode=ACTIVATION_CODE \
     --set eks.activationId=ACTIVATION_ID \
     --set eks.agentRegion=aws-region
   ```

   如果在上一步中使用了 AWS Management Console，请使用从上一步复制的已填充这些值的命令。

2. 检查已安装 eks-connector 部署的运行状况，并等待 Amazon EKS 中已注册集群的状态变为 ACTIVE。

YAML manifest

通过将 Amazon EKS Connector 清单文件应用于 Kubernetes 集群来完成连接。为此，必须使用前面描述的方法。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. 下载 Amazon EKS Connector YAML 文件。

   ```
   curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/
   eks-connector/latest/eks-connector.yaml
   ```

2. 编辑 Amazon EKS Connector YAML 文件，将 %AWS_REGION%、%EKS_ACTIVATION_ID%、%EKS_ACTIVATION_CODE% 的所有引用替换为上一步输出的 aws-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。

3. 在可以访问集群的终端中，可以运行以下命令应用更新的清单文件：

```
kubectl apply -f eks-connector.yaml
```

4. 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`。

5. （可选）向集群添加标签。有关更多信息，请参阅为您的 Amazon EKS 资源添加标签。

后续步骤

如果您对这些步骤有任何疑问，请参阅 排查 Amazon EKS Connector 中的问题。

要向其他 IAM 主体授予对 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 IAM 主体授予查看集群上的 Kubernetes 资源的访问权限。

向 IAM 主体授予查看集群上的 Kubernetes 资源的访问权限

授权 IAM 主体访问 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 AWS Management Console 所用的 IAM 主体必须满足以下要求：

- 它必须具有 `eks:AccessKubernetesApi` IAM 权限。
- Amazon EKS Connector 服务账户应能够模拟集群中的 IAM 主体。这让 Amazon EKS Connector 将 IAM 主体映射到 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 https://s3.us-west-2.amazonaws.com/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 主体的 Amazon 资源名称（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
```

要使某个 IAM 主体在 Amazon EKS 控制台上查看 Kubernetes 资源，该主体必须与 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 主体以访问连接的集群

1. 您可以下载以下任一示例清单文件，分别创建 `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` 集群角色允许访问控制台中可视化的所有命名空间和资源。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 https://s3.us-west-2.amazonaws.com/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 https://s3.us-west-2.amazonaws.com/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 主体的 Amazon 资源名称（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。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`

此过程包括两个步骤：向 Amazon EKS 取消注册集群，在集群中卸载 eks-connector 代理。

要取消注册 Kubernetes 集群

AWS CLI

先决条件

- 必须安装 AWS CLI。要进行安装或升级，请参阅安装 AWS 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
```

AWS 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。

eksctl

先决条件

- 必须安装 `eksctl` 版本 `0.68` 或更高版本。要安装或对其升级，请参阅 开始使用 Amazon EKS – `eksctl`。
- 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。

```
eksctl deregister cluster --name my-cluster
```

要清除 Kubernetes 集群中的资源

Helm

- 运行以下命令来卸载代理。

```
helm -n eks-connector uninstall eks-connector
```

YAML manifest

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 主体创建了 `clusterrole` 或 `clusterrolebindings` 来访问集群，请确保从 Kubernetes 集群中删除它们。

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和变通方法。

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

检查 Amazon EKS Connector 的状态。

```
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 AWS 账户和 AWS 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 集群注册到哪个 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'
```

- 使用以下命令可确定运行 Amazon EKS Connector 的节点名称。

```
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节点的信息。

```
kubectl get nodes -o wide --show-labels
```

Helm 问题：403 Forbidden

如果在运行 helm install 命令时收到以下错误：

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

您可以运行以下行来修复它：

```
docker logout public.ecr.aws
```

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 AWS。对于已注册的集群，Pending 状态表示未成功建立连接。要解决此问题，请确保您已将清单应用到目标 Kubernetes 集群。如果将其应用于集群，但集群仍处于 Pending 状态，则 `eks-connector` statefulset 可能不正常。要排查此问题，请参阅本主题中的Amazon EKS Connector Pods 处于崩溃循环 。

控制台错误：在集群范围内 User “system:serviceaccount:eks-connector:eks-connector” can't impersonate resource “users” in API group “”

Amazon EKS Connector 使用 Kubernetes 用户模拟代表 AWS Management Console 中的 IAM 主体进行操作。对于从 AWS `eks-connector` 服务账户访问 Kubernetes API 的每个主体，必须授予服务账户使用 IAM ARN 作为其 Kubernetes 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 AWS 账户 *111122223333* 的 IAM 用户 *john* 映射到 Kubernetes 用户。IAM 最佳实践建议您向角色而不是用户授予权限。

```
arn:aws:iam::111122223333:user/john
```

- 来自 AWS 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 AWS 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 IAM 主体授予查看集群上的 Kubernetes 资源的访问权限。确保在模板中将 %IAM_ARN% 替换为 AWS Management Console IAM 主体的 IAM ARN。

控制台错误：在集群范围内 [...] is forbidden: User [...] cannot list resource “[...] in API group”

考虑以下问题。Amazon EKS Connector 已成功模拟目标 Kubernetes 集群中请求的 AWS Management Console IAM 主体。但模拟主体没有 Kubernetes API 操作的 RBAC 权限。

要解决此问题，有两种方法可以向其他用户授予权限。如果您之前通过 Helm 图表安装了 eks-connector，则可以通过运行以下命令轻松授予用户访问权限。将 userARN1 和 userARN2 替换为 IAM 角色的 ARN 列表，以授予查看 Kubernetes 资源的访问权限：

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

或者，作为集群管理员，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 IAM 主体授予查看集群上的 Kubernetes 资源的访问权限。

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导致：

- 目标集群中的 Amazon EKS Connector 运行状况不佳。
- 目标集群与 AWS 区域之间的连接不佳或连接中断。

要解决此问题，请查看 Amazon EKS Connector 日志。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 AWS 区域 区域。

Amazon EKS Connector Pods 处于崩溃循环

可导致 Amazon 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。检查 Amazon EKS Connector Pod 的状态。

```
kubectl get pods -n eks-connector
```

示例输出如下。

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

如果您的输出与之前的输出类似，请参阅 检查 Amazon EKS Connector 的日志 排查问题。

Failed to initiate eks-connector: InvalidActivation

初次启动 Amazon EKS Connector 时，它会向 Amazon Web Services 注册 `activationId` 和 `activationCode`。注册可能会失败，进而可能导致 `connector-init` 容器崩溃，并显示类似以下错误。

```
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。
- 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三天内激活 Amazon EKS Connector。要解决此问题，请确保在到期日期和时间之前将 Amazon 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。

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

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。

```
{
    "cluster": {
        "name": "my-cluster",
        "arn": "arn:aws:eks:region:111122223333: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::111122223333: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。

集群节点缺少出站连接

为正常工作，Amazon EKS Connector 需要到多个 AWS 端点的出站连接。如果没有到目标 AWS 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。

Amazon EKS Connector Pods 处于 **ImagePullBackOff** 状态

如果您运行 get pods 命令且 Pods 处于 ImagePullBackOff 状态，则其无法正常工作。如果 Amazon EKS Connector Pods 处于 ImagePullBackOff 状态，则其无法正常工作。检查 Amazon EKS Connector Pods 的状态。

```
kubectl get pods -n eks-connector
```

示例输出如下。

```
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 Gallery 的映像。目标 Kubernetes 集群可能无法从 Amazon ECR Public Gallery 提取映像。解决 Amazon ECR Public Gallery 映像提取问题，或考虑映像您选择的私有容器注册表中的映像。

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 AWS Systems Manager（Systems Manager）代理。Amazon 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 AWS 断开连接。Systems Manager 代理是一个开源项目。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 AWS 区域 的出站连接。AWS 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 AWS 的连接。集群注册完成后，AWS 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 AWS？

答：Amazon EKS Connector 会发送在 AWS 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 AWS 时才会收集或发送此数据。默认情况下，AWS 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 AWS 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 Amazon EKS 服务可以位于任何 AWS 公共商业 AWS 区域。这适用于从集群到目标 AWS 区域的有效网络连接。我们建议您选择距集群位置最近的 AWS 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 AWS 区域（即 `ap-northeast-1` AWS 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 AWS 区域（中国或 GovCloud AWS 区域除外）中的 Amazon EKS。

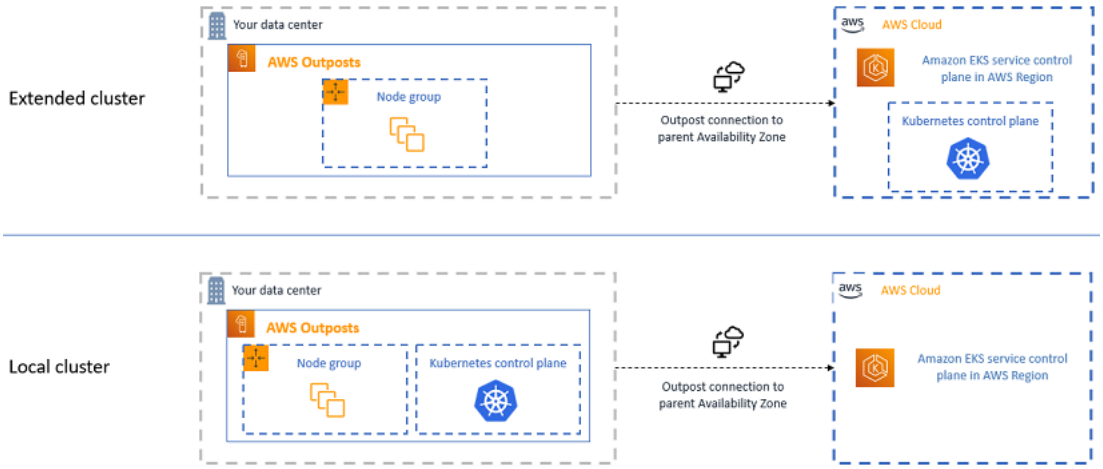
AWS Outposts 上的 Amazon EKS

您可以使用 Amazon EKS 在 AWS Outposts 上运行本地 Kubernetes 应用程序。您可以通过以下方式在 Outpost 上部署 Amazon EKS：

- 扩展集群 - 在 AWS 区域 中运行 Kubernetes 控制面板，在您的 Outpost 上运行节点。
- 本地集群 - 在您的 Outpost 上运行 Kubernetes 控制面板和节点。

对于这两个部署选项，Kubernetes 控制面板完全由 AWS 托管。您可以使用在云中使用的相同 Amazon EKS API、工具和控制台，在 Outpost 上创建和运行 Amazon EKS。

下图显示了这些部署选项。



何时使用每个部署选项

本地和扩展集群都是通用部署选项，可以用于一系列应用程序。

本地集群使您能在 Outpost 上本地运行整个 Amazon EKS 集群。此选项可以降低因临时断开与云的网络连接而造成的应用程序停机风险。光纤中断或天气事件可能会导致网络断开连接。由于整个 Amazon EKS 集群在 Outpost 上本地运行，应用程序仍然可用。您可以在断开与云的网络连接期间执行集群操作。有关更多信息，请参阅为网络断开连接做好准备。如果您担心从 Outpost 到父 AWS 区域 的网络连接质量，并且需要在整个网络断开连接期间保持高可用性，请使用本地集群部署选项。

扩展集群使您能够节省 Outpost 上的容量，因为 Kubernetes 控制面板在父 AWS 区域 中运行。如果您能投资建立从 Outpost 到 AWS 区域 的可靠、冗余网络连接，则此选项很合适。网络连接的质量对于此选项至关重要。Kubernetes 处理 Kubernetes 控制面板和节点之间网络断开连接的方式可能会导

致应用程序停机。有关 Kubernetes 的行为的更多信息，请参阅 Kubernetes 文档中的计划、抢占和驱逐。

部署选项比较

下表比较了两个选项之间的差异。

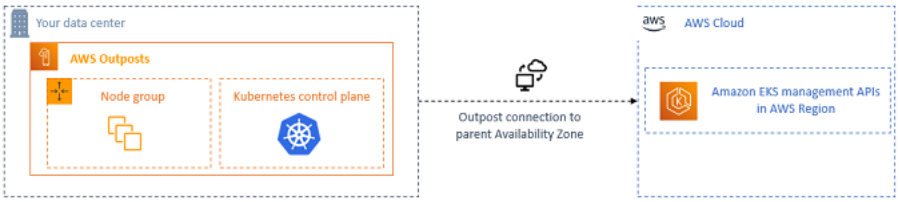
功能	扩展集群	本地集群
Kubernetes 控制面板位置	AWS 区域	Outpost
Kubernetes 控制面板账户	AWS 账户	您的账户
区域可用性	请参阅服务端点	美国东部（俄亥俄州）、美国东部（弗吉尼亚州北部）、美国西部（北加利福尼亚）、美国西部（俄勒冈州）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、亚太地区（东京）、加拿大（中部）、欧洲地区（法兰克福）、欧洲地区（爱尔兰）、欧洲地区（伦敦）、中东（巴林）和南美洲（圣保罗）。
Kubernetes 次要版本	支持的 Amazon EKS 版本。	支持的 Amazon EKS 版本。
平台版本	请参阅 Amazon EKS 平台版本	请参阅 Amazon EKS 本地集群平台版本
Outpost 外形规格	Outpost 机架	Outpost 机架
用户接口	AWS Management Console、AWS CLI、Amazon EKS API、`eksctl`、AWS CloudFormation 和 Terraform	AWS Management Console、AWS CLI、Amazon EKS API、`eksctl`、AWS CloudFormation 和 Terraform

功能	扩展集群	本地集群
托管策略	AmazonEKSClusterPolicy 和 AmazonEKSServiceRolePolicy	AmazonEKSLocalOutpostClusterPolicy 和 AmazonEKSLocalOutpostServiceRolePolicy
集群 VPC 和子网	请参阅 Amazon EKS VPC 和子网要求和注意事项	请参阅 Amazon EKS 本地集群 VPC 及子网的要求和注意事项
集群终端节点访问	公共或私有，或者两者兼而有之	仅限私有
Kubernetes API 服务器身份验证	AWS Identity and Access Management (IAM) 和 OIDC	IAM 和 x.509 证书
节点类型	仅自行管理	仅自行管理
节点计算类型	Amazon EC2 按需	Amazon EC2 按需
节点存储类型	Amazon EBS gp2 和本地 NVMe SSD	Amazon EBS gp2 和本地 NVMe SSD
Amazon EKS 优化版 AMI	Amazon Linux、Windows 和 Bottlerocket	仅限 Amazon Linux
IP 版本	仅限 IPv4	仅限 IPv4
附加组件	Amazon EKS 附加组件或自行管理的附加组件	自行管理的附加组件
默认容器网络接口	Amazon VPC CNI plugin for Kubernetes	Amazon VPC CNI plugin for Kubernetes
Kubernetes 控制面板日志	Amazon CloudWatch Logs	Amazon CloudWatch Logs
负载均衡	使用 AWS Load Balancer Controller 仅预调配应用程序负载均衡器（无网络负载均衡器）	使用 AWS Load Balancer Controller 仅预调配应用程序负载均衡器（无网络负载均衡器）

功能	扩展集群	本地集群
秘密信封加密	请参阅 在现有集群中启用密钥加密	不支持
服务账户的 IAM 角色	请参阅 服务账户的 IAM 角色	不支持
故障排除	请参阅 Amazon EKS 故障排除	请参阅 对 AWS Outposts 上的 Amazon EKS 的本地集群进行故障排除

AWS Outposts 上的 Amazon EKS 的本地集群

您可以使用本地集群在 AWS Outposts 上本地运行您的整个 Amazon EKS 集群。这有助于降低因临时断开与云的网络连接而造成的应用程序停机风险。光纤中断或天气事件可能会导致此类连接断开的情况。由于整个 Kubernetes 集群在 Outpost 上本地运行，应用程序仍然可用。您可以在断开与云的网络连接期间执行集群操作。有关更多信息，请参阅为网络断开连接做好准备。下图显示了本地集群部署。



本地集群通常可用于与 Outpost 机架配合使用。

支持的 AWS 区域

您可以在以下 AWS 区域 创建本地集群：美国东部（俄亥俄州）、美国东部（弗吉尼亚州北部）、美国西部（北加利福尼亚）、美国西部（俄勒冈州）、亚太地区（首尔）、亚太地区（新加坡）、亚太地区（悉尼）、亚太地区（东京）、加拿大（中部）、欧洲地区（法兰克福）、欧洲地区（爱尔兰）、欧洲地区（伦敦）、中东（巴林）和南美洲（圣保罗）。有关受支持的功能的详细信息，请参阅 部署选项比较。

主题

- 在 Outpost 上创建本地集群
- Amazon EKS 本地集群平台版本
- Amazon EKS 本地集群 VPC 及子网的要求和注意事项

- 为网络断开连接做好准备
- 容量注意事项
- 对 AWS Outposts 上的 Amazon EKS 的本地集群进行故障排除

在 Outpost 上创建本地集群

本主题概述了在 Outpost 上运行本地集群时需要考虑的事项。本主题还提供了有关如何在 Outpost 上部署本地集群的说明。

注意事项

> **Important**
>
> - 相关的 Amazon EKS 文档中没有重复这些注意事项。若其他 Amazon EKS 文档主题与本文的考虑事项冲突，则遵从本文的考虑事项。
> - 这些注意事项可能会经常更改。因此，我们建议您定期查看本主题。
> - 很多注意事项与在 AWS Cloud 上创建集群的注意事项不同。

- 本地集群仅支持 Outpost 机架。单个本地集群可跨构成单个逻辑 Outpost 的多个物理 Outpost 机架运行。单个本地集群不能跨多个逻辑 Outpost 运行。每个逻辑 Outpost 都有一个 Outpost ARN。
- 本地集群运行和管理 Outpost 上您的账户中的 Kubernetes 控制面板。您不能在 Kubernetes 控制面板实例上运行工作负载或对 Kubernetes 控制面板组件进行修改。这些节点由 Amazon EKS 服务管理。对 Kubernetes 控制面板的更改不会在自动 Amazon EKS 管理操作（如修补）期间一直持续存在。
- 本地集群支持自行管理的附加组件和自行管理的 Amazon Linux 节点组。Amazon VPC CNI plugin for Kubernetes、`kube-proxy` 和 CoreDNS 附加组件将自动安装在本地集群上。
- 本地集群需要使用 Outpost 上的 Amazon EBS。您的 Outpost 必须拥有可用于 Kubernetes 控制面板存储的 Amazon EBS。
- 本地集群使用 Outpost 上的 Amazon EBS。您的 Outpost 必须拥有可用于 Kubernetes 控制面板存储的 Amazon EBS。Outpost 仅支持 Amazon EBS gp2 卷。
- 使用 Amazon EBS CSI 驱动程序支持 Amazon EBS 支持的 Kubernetes `PersistentVolumes`。

先决条件

- 熟悉 Outpost 部署选项、容量注意事项 和 Amazon EKS 本地集群 VPC 及子网的要求和注意事项。
- 一个现有的 Outpost。有关更多信息，请参阅什么是 AWS Outposts。
- 计算机或 AWS 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 `1.29`，则可以将 `kubectl` 的 `1.28`、`1.29` 或 `1.30` 版本与之配合使用。要安装或升级 `kubectl`，请参阅 安装或更新 `kubectl`。
- 在您的设备或 AWS CloudShell 上安装和配置了 AWS Command Line Interface（AWS CLI）的版本 `2.12.3` 或更高版本，或版本 `1.27.160` 或更高版本。要查看当前版本，请使用 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**。软件包管理器（如 `yum`、`apt-get` 或适用于 macOS 的 Homebrew）通常比 AWS CLI 的最新版本落后几个版本。要安装最新版本，请参阅《AWS Command Line Interface 用户指南》中的安装、更新和卸载 AWS CLI，以及使用 aws configure 快速配置。AWS CloudShell 中安装的 AWS CLI 版本也可能比最新版本落后几个版本。如需更新，请参阅《AWS CloudShell 用户指南》中的将 AWS CLI 安装到主目录。
- 具有 `create` 和 `describe` Amazon EKS 集群权限的 IAM 主体（用户或角色）。有关更多信息，请参阅在 Outpost 上创建本地 Kubernetes 集群 和列出或描述所有集群。

创建本地 Amazon EKS 集群后，创建集群的 IAM 主体将永久添加。主体将作为管理员专门添加到 Kubernetes RBAC 授权表。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，仅创建服务器的主体可以使用 `kubectl` 调用 Kubernetes API 服务器。如果使用控制台创建集群，请确保在集群上运行 `kubectl` 命令时，相同的 IAM 凭证位于 AWS SDK 凭证链中。创建集群后，您可以向其他 IAM 主体授予对集群的访问权限。

创建本地 Amazon EKS 本地集群

您可以使用 `eksctl`、AWS Management Console、AWS CLI、Amazon EKS API、AWS SDK、AWS CloudFormation 或 Terraform 创建本地集群。

1. 创建本地集群。

   eksctl

   先决条件

   您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

使用 **eksctl** 创建集群

1. 将后续内容复制到您的设备。替换以下值，然后运行修改后的命令以创建 outpost-control-plane.yaml 文件：
   - 将 *region-code* 替换为您要在其中创建集群的受支持的 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - 将 *vpc-ExampleID1* 和 *subnet-ExampleID1* 替换为现有 VPC 和子网的 ID。VPC 和子网必须满足 Amazon EKS 本地集群 VPC 及子网的要求和注意事项 中的要求。
   - 将 *uniqueid* 替换为您的 Outpost 的 ID。
   - 将 *m5.large* 替换为您的 Outpost 上可用的实例类型。在选择实例类型之前，请参阅 容量注意事项。将部署三个控制面板实例。您无法更改此数字。

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.24"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws:outposts:region-code:111122223333:outpost/
op-uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

有关所有可用选项和默认设置的完整列表，请参阅 AWS Outposts 支持和 eksctl 文档中的配置文件架构。

2. 使用您在上一步中创建的配置文件创建集群。eksctl 将在您的 Outpost 上创建一个 VPC 和一个子网，以在其中部署集群。

```
eksctl create cluster -f outpost-control-plane.yaml
```

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。

Eksctl 自动为创建集群的 IAM 主体（用户或角色）创建访问条目，并授予 IAM 主体对集群上 Kubernetes 对象的管理员权限。如果您不希望集群创建者拥有集群上 Kubernetes 对象的管理员访问权限，请在之前的配置文件中添加以下文本：**bootstrapClusterCreatorAdminPermissions: false**（与 metadata、vpc 和 outpost 的级别相同）。如果您添加了该选项，则在创建集群后，您需要为至少一个 IAM 主体创建访问条目，否则任何 IAM 主体将无法访问集群上的 Kubernetes 对象。

AWS Management Console

先决条件

满足 Amazon EKS 要求的现有 VPC 和子网。有关更多信息，请参阅 Amazon EKS 本地集群 VPC 及子网的要求和注意事项。

使用 AWS Management Console 创建集群

1. 如果您已经拥有本地集群 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。

   a. 运行以下命令以创建 IAM 信任策略 JSON 文件。

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

   b. 创建 Amazon EKS 集群 IAM 角色。要创建 IAM 角色，必须为正在创建角色的 IAM 主体分配 iam:CreateRole 操作（权限）。

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

   c. 将名为 AmazonEKSLocalOutpostClusterPolicy 的 Amazon EKS 托管 IAM 策略附加到角色。要将 IAM 策略附加到某个 IAM 主体，必须为附加该策略的主体分配以下 IAM 操作（权限）之一：iam:AttachUserPolicy 或 iam:AttachRolePolicy。

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

2. 从以下位置打开 Amazon EKS 控制台：https://console.aws.amazon.com/eks/home#/clusters。
3. 在控制台屏幕的顶部，确保您已选择受支持的 AWS 区域。
4. 请选择 Add cluster（添加集群），然后选择 Create（创建）。
5. 在 Configure cluster（配置集群）页面上，为以下字段输入或选择值：

- Kubernetes 控制面板位置 – 选择 AWS Outposts。
- Outpost ID - 选择您要在其上创建控制面板的 Outpost 的 ID。
- Instance type（实例类型）– 选择一个实例类型。仅显示您的 Outpost 中可用的实例类型。在下拉列表中，每种实例类型都将描述为实例类型推荐多少个节点。在选择实例类型之前，请参阅 容量注意事项。所有副本都将使用相同实例类型进行部署。在创建您的集群后，您将无法更改实例类型。将部署三个控制面板实例。您无法更改此数字。
- Name（名称）– 集群的名称。它必须在您的 AWS 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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
- Kubernetes 版本 – 选择要用于集群的 Kubernetes 版本。建议选择最新版本，除非您需要使用早期版本。
- Cluster service role（集群服务角色）– 选择您在上一步创建的 Amazon EKS 集群 IAM 角色，以允许 Kubernetes 控制面板管理 AWS 资源。
- Kubernetes 集群管理员访问权限 – 如果您希望创建集群的 IAM 主体（角色或用户）拥有集群上 Kubernetes 对象的管理员访问权限，请接受默认设置（允许）。Amazon EKS 为 IAM 主体创建访问条目，并向集群管理员授予访问条目的权限。有关访问条目的更多信息，请参阅 管理访问条目。

  如果您希望与创建集群的主体不同的 IAM 主体拥有 Kubernetes 集群对象的管理员访问权限，请选择“不允许”选项。创建集群后，任何拥有创建访问条目的 IAM 权限的 IAM 主体都可以为需要访问 Kubernetes 集群对象的任何 IAM 主体添加访问条目。如需了解所需的 IAM 权限，请参阅《服务授权参考》中的 Amazon Elastic Kubernetes Service 定义的操作。如果您选择“不允许”选项并且不创建任何访问条目，则没有 IAM 主体有权访问集群上的 Kubernetes 对象。
- 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。

  完成此页面后，请选择下一步。

6. 在 Specify networking (指定联网) 页面上，为以下字段选择值：
   - VPC – 选择现有 VPC。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。您的 VPC 必须满足 VPC 要求和注意事项 中的要求。
   - 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。您选择的子网必须满足 子网要求和注意事项 中的要求。

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。Amazon EKS 会自动创建安全组，以实现集群与 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 Amazon EKS 安全组要求和注意事项。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。要使本地主机与集群端点进行通信，您必须允许来自集群安全组的入站流量。对于没有传入和传出互联网连接（也被称为私有集群）的集群，您必须执行以下操作之一：

- 添加与所需 VPC 端点关联的安全组。有关所需端点的更多信息，请参阅 子网对 AWS 服务 的访问 中的接口 VPC 端点。
- 修改 Amazon EKS 创建的安全组，以允许来自与 VPC 端点关联的安全组的流量。

完成此页面后，请选择下一步。

7. 在配置可观测性页面上，您可以选择要开启的指标和控制面板日志记录选项。默认情况下，每种日志类型都处于关闭状态。
   - 有关 Prometheus 指标选项的更多信息，请参阅 创建集群时开启 Prometheus 指标。
   - 有关控制面板日志记录选项的更多信息，请参阅 Amazon EKS 控制面板日志记录。

   完成此页面后，请选择下一步。
8. 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）。

   集群预配置需要几分钟时间。

2. 在创建您的集群后，您可以查看已创建的 Amazon EC2 控制面板实例。

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

示例输出如下。

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

每个实例都将受到 node-role.eks-local.amazonaws.com/control-plane 的污染，这样就不会在控制面板实例上安排任何工作负载。有关污点的更多信息，请参阅 Kubernetes 文档中的污点和容忍度。Amazon EKS 将持续监控本地集群的状态。我们将执行自动管理操作，如安全补丁和修复运行状况不佳的实例。当本地集群与云断开连接时，我们会完成操作，以确保在重新连接后将集群修复到运行状况正常的状态。

3. 如果您使用 eksctl 创建集群，则可以跳过此步骤。eksctl 会为您完成此步骤。通过向 kubectl config 文件添加新上下文来启用 kubectl 与您的集群通信。有关如何创建或更新文件的说明，请参阅 为 Amazon EKS 集群创建或更新 kubeconfig 文件。

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

示例输出如下。

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

4. 要连接到您的本地集群的 Kubernetes API 服务器，您必须有权访问子网的本地网关，或从 VPC 内部进行连接。有关将 Outpost 机架连接到您的本地网络的更多信息，请参阅《AWS Outposts 用户指南》中的机架的本地网关的工作原理。如果您使用直接 VPC 路由且 Outpost 子网具有通向您的本地网关的路由，则 Kubernetes 控制面板实例的私有 IP 地址将在您的本地网络内自动广播。本地集群的 Kubernetes API 服务器端点托管在 Amazon Route 53（Route 53）中。API 访问端点可由公有 DNS 服务器解析至 Kubernetes API 服务器的私有 IP 地址。

   本地集群的 Kubernetes 控制面板实例将配置具有固定私有 IP 地址的静态弹性网络接口，这些固定私有 IP 地址在整个集群生命周期中都不会更改。在网络连接断开期间，与 Kubernetes API 服务器互动的设备可能无法连接到 Route 53。如果是这种情况，我们建议使用静态私有 IP 地址配置 /etc/hosts 以实现持续运行。我们还建议设置本地 DNS 服务器，并将其连接到 Outpost。有关更多信息，请参阅 AWS Outposts 文档。运行以下命令以确认已建立与集群的通信。

```
kubectl get svc
```

示例输出如下。

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

5. （可选）当本地集群与 AWS Cloud 断开连接时，对其进行身份验证测试。有关说明，请参阅为网络断开连接做好准备。

内部资源

Amazon EKS 将在您的集群上创建以下资源。这些资源将供 Amazon EKS 内部使用。为使您的集群正常运行，请不要编辑或修改这些资源。

- 以下映像 Pods：
  - `aws-iam-authenticator-node-hostname`
  - `eks-certificates-controller-node-hostname`
  - `etcd-node-hostname`
  - `kube-apiserver-node-hostname`
  - `kube-controller-manager-node-hostname`
  - `kube-scheduler-node-hostname`
- 以下自行管理的附加组件：
  - `kube-system/coredns`
  - `kube-system/kube-proxy`（在您添加第一个节点之前不会创建）
  - `kube-system/aws-node`（在您添加第一个节点之前不会创建）。本地集群将使用 Amazon VPC CNI plugin for Kubernetes 插件进行集群联网。不要更改控制面板实例（名为 `aws-node-controlplane-*` 的容器组 (pod)）的配置。您可以使用一些配置变量在插件创建新网络接口时更改默认值。有关更多信息，请参阅 GitHub 上的文档。
- 以下服务：
  - `default/kubernetes`
  - `kube-system/kube-dns`
- 名为 `eks.system` 的 `PodSecurityPolicy`
- 名为 `eks:system:podsecuritypolicy` 的 `ClusterRole`
- 名为 `eks:system` 的 `ClusterRoleBinding`
- 默认值 PodSecurityPolicy
- 除集群安全组以外，Amazon EKS 还会在您名为 `eks-local-internal-do-not-use-or-edit-cluster-name-uniqueid` 的 AWS 账户 中创建一个安全组。此安全组允许流量在控制面板实例上运行的 Kubernetes 组件之间自由流动。

向您建议的后续步骤：

- 向创建集群的 IAM 主体授予所需的权限，以便在 AWS Management Console 中查看 Kubernetes 资源
- 授予 IAM 实体访问您的集群的权限。如果您希望实体在 Amazon EKS 控制台中查看 Kubernetes 资源，请向实体授予 所需的权限。
- 为集群配置日志记录
- 熟悉网络断开连接期间发生的情况。
- 将节点添加到集群
- 考虑为您的 `etcd` 制定备份计划。Amazon EKS 不支持为本地集群自动备份和恢复 `etcd`。有关更多信息，请参阅 Kubernetes 文档中的备份 `etcd` 集群。两个主要选项使用 `etcdctl` 自动拍摄快照或使用 Amazon EBS 存储卷备份。

Amazon EKS 本地集群平台版本

本地集群平台版本表示 AWS Outposts 上的 Amazon EKS 集群的功能。这些版本包括在 Kubernetes 控制面板上运行的组件，启用了哪些 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平台版本。不同 Kubernetes 次要版本的平台版本是独立的。本地集群和云中的 Amazon EKS 集群的平台版本是独立的。

当新的 Kubernetes 次要版本可用于本地集群（如 `1.28`）时，该 Kubernetes 次要版本的初始平台版本将从 `eks-local-outposts.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。

当新的本地集群平台版本可用于次要版本时：

- 平台版本号将递增 (`eks-local-outposts.n+1`)。
- Amazon EKS 会自动将所有现有本地集群更新到其对应的 Kubernetes 次要版本的最新平台版本。现有平台版本的自动更新将以增量方式推出。推出过程可能需要一段时间。如果您即刻就需要最新的平台版本功能，我们建议您创建新的本地集群。
- 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。对于单个 Kubernetes 次要版本，在 Kubernetes 控制面板与节点 AMI 之间的所有补丁版本都兼容。

新平台版本不会引发破坏性的更改或导致服务中断。

本地集群始终使用指定 Kubernetes 版本的最新可用平台版本 (`eks-local-outposts.n`) 创建。

下列各表列出了当前和最新的平台版本。

Kubernetes 版本 1.28

所有 1.28 平台版本都启用了以下准入控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Defaul
和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.28.6	eks-local-outposts.5	已将 Bottlerocket 版本更新为 v1.19.3（其中包含最新的错误修复），以支持 Outposts 中的本地启动。	2024 年 4 月 18 日
1.28.6	eks-local-outposts.4	新的平台版本（包括安全修复和增强功能）。已恢复 Outposts 中的支持或本地启动。出于兼容性考虑，已将 Bottlerocket 版本降级为 v1.15.1。	2024 年 4 月 2 日
1.28.6	eks-local-outposts.3	新的平台版本（包括安全修复和增强功能）。	2024 年 3 月 22 日
1.28.6	eks-local-outposts.2	具有安全修复和增强功能 kube-proxy 的新平台版本已更新为 v1.28.6。AWSIAM 身份验证器已更新为 v0.6.17。出于兼容性考虑，已将适用于 Kubernetes 的 Amazon VPC CNI 插件降级为 v1.13.2。已将 Bottlerocket 版本更新为 v1.19.2。	2024 年 3 月 8 日
1.28.1	eks-local-outposts.1	Outpost 上适用于本地 Amazon EKS 集群的 Kubernetes 版本 v1.28 的初始版本	2023 年 10 月 4 日

Kubernetes 版本 1.27

所有 1.27 平台版本都启用了以下准入控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Defaul和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.27.10	eks-local-outposts.5	具有安全修复和增强功能的新平台。	2024 年 4 月 2 日
1.27.10	eks-local-outposts.4	已将具有安全修复和增强功能 kube-proxy 的新平台更新为 v1.27.10。AWSIAM 身份验证器已更新为 v0.6.17。已将 Bottlerocket 版本更新为 v1.19.2。	2024 年 3 月 22 日
1.27.3	eks-local-outposts.3	包含安全修复和增强功能的新平台版本。kube-proxy 升级到 v1.27.3。适用于 Kubernetes 的 Amazon VPC CNI 插件升级到 v1.13.2。	2023 年 7 月 14 日
1.27.1	eks-local-outposts.2	已将 CoreDNS 映像更新为 v1.10.1	2023 年 6 月 22 日
1.27.1	eks-local-outposts.1	Outpost 上适用于本地 Amazon EKS 集群的 Kubernetes 版本 1.27 的初始版本。	2023 年 5 月 30 日

Kubernetes 版本 1.26

所有 1.26 平台版本都启用了以下准入控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Defaul和 ValidatingAdmissionWebhook。

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.26.13	eks-local-outposts.5	已将具有安全修复和增强功能 kube-proxy 的新平台版本更新为 v1.26.13。AWSIAM 身份验证器已更新为 v0.6.17。已将 Bottlerocket 版本更新为 v1.19.2。	2024 年 3 月 22 日

Kubernetes 版本 **1.25**

所有 1.25 平台版本都启用了以下准入控制器：CertificateApproval、CertificateSigning、CertificateSubjectRestriction、Defaul

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.25.16	eks-local-outposts.7	已将具有安全修复和增强功能 kube-proxy 的新平台版本更新为 v1.25.16。AWSIAM 身份验证器已更新为 v0.6.17。已将 Bottlerocket 版本更新为 v1.19.2。	2024 年 3 月 22 日
1.25.11	eks-local-outposts.6	包含安全修复和增强功能的新平台版本。kube-proxy 升级到 v1.25.11。适用于 Kubernetes 的 Amazon VPC CNI 插件升级到 v1.13.2。	2023 年 7 月 14 日
1.25.9	eks-local-outposts.5	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 13 日
1.25.6	eks-local-outposts.4	已将 Bottlerocket 版本更新为 1.13.2	2023 年 5 月 2 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.25.6	eks-local-outposts.3	已将 Amazon EKS 控制面板实例操作系统更新为 Bottlerocket 版本 v1.13.1，并且适用于 Kubernetes 的 Amazon VPC CNI 插件已更新为版本 v1.12.6。	2023 年 4 月 14 日
1.25.6	eks-local-outposts.2	改进了 Kubernetes 控制面板实例的诊断收集。	2023 年 3 月 8 日
1.25.6	eks-local-outposts.1	Outpost 上适用于本地 Amazon EKS 集群的 Kubernetes 版本 1.25 的初始版本。	2023 年 3 月 1 日

Kubernetes 版本 **1.24**

所有 1.24 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionW

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.24.17	eks-local-outposts.7	已将具有安全修复和增强功能 kube-proxy 的新平台版本更新为 v1.25.16。AWSIAM 身份验证器已更新为 v0.6.17。已将 Bottlerocket 版本更新为 v1.19.2。	2024 年 3 月 22 日
1.24.15	eks-local-outposts.6	包含安全修复和增强功能的新平台版本。kube-proxy 升级到 v1.24.15。适用于 Kubernetes 的 Amazon VPC CNI 插件升级到 v1.13.2。	2023 年 7 月 14 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.24.13	eks-local-outposts.5	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 13 日
1.24.9	eks-local-outposts.4	已将 Bottlerocket 版本更新为 1.13.2	2023 年 5 月 2 日
1.24.9	eks-local-outposts.3	已将 Amazon EKS 控制面板实例操作系统更新为 Bottlerocket 版本 v1.13.1，并且适用于 Kubernetes 的 Amazon VPC CNI 插件已更新为版本 v1.12.6。	2023 年 4 月 14 日
1.24.9	eks-local-outposts.2	改进了 Kubernetes 控制面板实例的诊断收集。	2023 年 3 月 8 日
1.24.9	eks-local-outposts.1	Outpost 上适用于本地 Amazon EKS 集群的 Kubernetes 版本 1.24 的初始版本。	2023 年 1 月 17 日

Kubernetes 版本 **1.23**

所有 1.23 平台版本都启用了以下准入控制器：DefaultStorageClass、DefaultTolerationSeconds、LimitRanger、MutatingAdmissionW

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
1.23.17	eks-local-outposts.6	新的平台版本（包括安全修复和增强功能）。	2023 年 7 月 13 日
1.23.17	eks-local-outposts.5	已将具有安全修复和增强功能 kube-proxy 的新平台版本更新为	2023 年 7 月 6 日

Kubernetes 版本	Amazon EKS 平台版本	发布说明	发行日期
		v1.23.17。已将 Bottlerocket 版本更新为 v1.14.1。	
1.23.15	eks-local-outposts.4	已将 Amazon EKS 控制面板实例操作系统更新为 Bottlerocket 版本 v1.13.1，并且适用于 Kubernetes 的 Amazon VPC CNI 插件已更新为版本 v1.12.6。	2023 年 4 月 14 日
1.23.15	eks-local-outposts.3	改进了 Kubernetes 控制面板实例的诊断收集。	2023 年 3 月 8 日
1.23.15	eks-local-outposts.2	Outpost 上适用于本地 Amazon EKS 集群的 Kubernetes 版本 1.23 的初始版本。	2023 年 1 月 17 日

Amazon EKS 本地集群 VPC 及子网的要求和注意事项

在您创建本地集群时，您将指定 VPC 以及在 Outpost 上运行的至少一个私有子网。本主题概述适用于本地集群的 VPC 和子网的要求和注意事项。

VPC 要求和注意事项

在您创建本地集群时，您指定的 VPC 必须满足以下要求和注意事项：

- 请确保 VPC 具有足够的 IP 地址用于本地集群、任何节点以及您要创建的其他 Kubernetes 资源。如果要使用的 VPC 没有足够的 IP 地址，请增加可用的 IP 地址数量。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 块关联到您的 VPC。集群最多可能需要 5 个小时的时间来识别与 VPC 关联的 CIDR 块。
- VPC 无法分配 IP 前缀或 IPv6 CIDR 块。由于这些限制，提高 Amazon EC2 节点的可用 IP 地址数量 和 集群、Pods 和 services 的 IPv6 地址 中包含的信息不适用于您的 VPC。

- VPC 已启用 DNS 主机名和 DNS 解析。如果没有这些功能，本地集群将无法创建，因此您需要启用这些功能，然后重新创建集群。有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 的 DNS 属性。
- 要通过本地网络访问本地集群，VPC 必须与 Outpost 的本地网关路由表关联。有关更多信息，请参阅《AWS Outposts 用户指南》中的 VPC 关联。

子网要求和注意事项

创建集群时，请指定至少一个私有子网。如果您指定多个子网，Kubernetes 控制面板实例将均匀分布在这些子网中。如果指定多个子网，则这些子网必须存在于同一 Outpost 上。此外，子网还必须具有适当的路由和安全组权限，以便相互通信。创建本地集群时，您指定的子网必须满足以下要求：

- 子网均位于同一逻辑 Outpost 上。
- 这些子网总共具有至少三个可用 IP 地址用于 Kubernetes 控制面板实例。如果指定三个子网，则每个子网必须具有至少一个可用 IP 地址。如果指定两个子网，则每个子网必须具有至少两个可用 IP 地址。如果指定一个子网，则该子网必须具有至少三个可用 IP 地址。
- 子网具有通向 Outpost 机架的本地网关的路由，用于通过您的本地网络访问 Kubernetes API 服务器。如果子网没有通向 Outpost 机架的本地网关的路由，则您必须从 VPC 中与 Kubernetes API 服务器通信。
- 子网必须使用基于 IP 地址的命名。Amazon EKS 不支持 Amazon EC2 基于资源的命名。

子网对 AWS 服务 的访问

Outpost 上本地集群的私有子网必须能够与区域性 AWS 服务 通信。您可以通过使用 NAT 网关进行出站互联网访问来实现此目的，或者，如果您想在 VPC 内保持所有流量的私密性，则使用接口 VPC 端点。

使用 NAT 网关

Outpost 上本地集群的私有子网必须具有关联的路由表，其中包含通向 NAT 网关（在 Outpost 的父可用区内的公有子网中）的路由。该公有子网必须具有一条通向互联网网关的路由。NAT 网关可以实现出站互联网访问，并防止从互联网到 Outpost 上实例进行未经请求的入站连接。

使用接口 VPC 端点

如果 Outpost 上本地集群的私有子网没有出站互联网连接，或者如果您想在 VPC 内保持所有流量的私密性，则在创建集群之前，您必须在区域性子网中创建以下接口 VPC 端点和网关端点。

终端节点	端点类型
com.amazonaws. *region-code* .ssm	接口
com.amazonaws. *region-code* .ssmmessages	接口
com.amazonaws. *region-code* .ec2messages	接口
com.amazonaws. *region-code* .ec2	接口
com.amazonaws. *region-code* .secretsmanager	接口
com.amazonaws. *region-code* .logs	接口
com.amazonaws. *region-code* .sts	接口
com.amazonaws. *region-code* .ecr.api	接口
com.amazonaws. *region-code* .ecr.dkr	接口
com.amazonaws. *region-code* .s3	Gateway

端点必须满足以下要求：

- 在位于 Outpost 父可用区的私有子网中创建
- 启用私有 DNS 名称
- 具有附加的安全组，该组允许来自私有 outpost 子网的 CIDR 范围的入站 HTTPS 流量。

创建端点会产生费用。有关更多信息，请参阅AWS PrivateLink 定价。如果您的 Pods 需要访问其他 AWS 服务，您需要创建额外的端点。有关端点的完整列表，请参阅 与 AWS PrivateLink 集成的 AWS 服务。

创建 VPC

您可以使用以下 AWS CloudFormation 模板之一，创建满足上述要求的 VPC：

- 模板 1 – 此模板会创建一个 VPC，该 VPC 具有一个位于 Outpost 的私有子网，以及一个位于 AWS 区域 的公有子网。该私有子网具有通过位于 AWS 区域 中公有子网中的 NAT 网关通向互联网的路由。此模板可用于在具有传出互联网访问权限的子网中创建本地集群。
- 模板 2 – 此模板会创建一个 VPC，该 VPC 具有一个位于 Outpost 的私有子网，以及在没有传入或传出互联网访问权限的子网（也称为私有子网）中创建本地集群所需的最小 VPC 端点集。

为网络断开连接做好准备

如果本地网络断开与 AWS Cloud 的连接，您可以继续在 Outpost 上使用本地 Amazon EKS 集群。本主题介绍如何针对网络断开连接的情况准备本地集群以及相关注意事项。

针对网络断开连接的情况准备本地集群时的注意事项：

- 本地集群可在临时、计划外网络断开连接期间实现稳定性和持续操作。AWS Outposts 仍是一个完全连接的产品，充当您的数据中心内 AWS Cloud 的扩展。当 Outpost 与 AWS Cloud 之间的网络断开连接，我们建议您尝试恢复连接。如需查看相关说明，请参阅《AWS Outposts 用户指南》中的 AWS Outposts 机架网络故障排除清单。有关如何对本地集群的问题进行故障排除的更多信息，请参阅 对 AWS Outposts 上的 Amazon EKS 的本地集群进行故障排除。
- Outpost 将发出一个 `ConnectedStatus` 指标，您可用于监控 Outpost 的连接状态。有关更多信息，请参阅《AWS Outposts 用户指南》中的 Outposts 指标。
- 本地集群使用 IAM 作为默认身份验证机制，该身份验证机制将 AWS Identity and Access Management 身份验证器用于 Kubernetes。IAM 在网络连接断开期间不可用。因此，本地集群支持使用 `x.509` 证书的替代身份验证机制，您可以使用该机制在网络连接断开期间连接到集群。有关如何获取 `x.509` 证书并将其用于您的集群的信息，请参阅 在网络断开连接期间对您的本地集群进行身份验证。
- 如果您在网络连接断开期间无法访问 Route 53，请考虑在您的本地环境中使用本地 DNS 服务器。Kubernetes 控制面板实例使用静态 IP 地址。您可以使用端点主机名和 IP 地址配置用于连接到您的集群的主机，作为使用本地 DNS 服务器的替代方法。有关更多信息，请参阅 AWS Outposts 用户指南中的 DNS。
- 如果您预计在网络连接断开期间应用程序流量会增加，则可以在连接到云时在您的集群中预调配备用计算容量。Amazon EC2 实例包含在 AWS Outposts 的价格中。因此，运行备用实例不会影响您的 AWS 用量成本。
- 在网络连接断开期间，要启用工作负载的创建、更新和扩缩操作，必须可以通过本地网络访问您的应用程序的容器映像，并且您的集群必须具有足够的容量。本地集群不会为您托管容器注册表。如果 Pods 之前已在节点上运行过，则容器映像将缓存在这些节点上。如果您通常从云中的 Amazon ECR

提取应用程序的容器映像，建议运行本地缓存或注册表。如果您需要在网络连接断开期间创建、更新和扩展工作负载资源的操作，那么本地缓存或注册表会很有帮助。

- 本地集群使用 Amazon EBS 作为持久卷的默认存储类，并使用 Amazon EBS CSI 驱动程序来管理 Amazon EBS 持久卷的生命周期。网络连接断开期间，由 Amazon EBS 备份的 Pods 将无法创建、更新或扩展。这是因为这些操作需要调用云端的 Amazon EBS API。如果您在本地集群上部署有状态工作负载，并且需要在网络断开连接期间执行创建、更新或扩缩操作，请考虑使用替代存储机制。
- 如果 AWS Outposts 无法访问相关 AWS 区域内 API（例如 Amazon EBS 或 Amazon S3 的 API），则无法创建或删除 Amazon EBS 快照。
- 将 ALB（Ingress）与 AWS Certificate Manager（ACM）集成时，证书会被推送并存储在 AWS Outposts ALB 计算实例的内存中。如果与 AWS 区域断开连接，当前的 TLS 终止将继续运行。在这种情况下，更改操作将失败（例如新的传入定义、新的基于 ACM 的证书 API 操作、ALB 计算扩展或证书轮换）。有关更多信息，请参阅《AWS Certificate Manager 用户指南》中的排查托管证书续订的问题。
- 在网络断开连接期间，Amazon EKS 控制面板日志将本地缓存在 Kubernetes 控制面板实例上。重新连接后，会将这些日志发送到父级 AWS 区域 中的 CloudWatch 日志。您可以使用 Prometheus、Grafana 或 Amazon EKS 合作伙伴解决方案，使用 Kubernetes API 服务器的指标端点或将 Fluent Bit 用于日志，以在本地监控集群。
- 如果您将 Outpost 上的 AWS Load Balancer Controller 用于应用程序流量，则在网络连接断开期间，前面带有 AWS Load Balancer Controller 的现有 Pods 将继续接收流量。在网络断开连接期间创建的新 Pods 不会接收流量，直到 Outpost 重新连接到 AWS Cloud 为止。考虑在连接到 AWS Cloud 时为您的应用程序设置副本计数，以满足您在网络断开连接期间的扩缩需求。
- Amazon VPC CNI plugin for Kubernetes 默认为辅助 IP 模式。它配置为 `WARM_ENI_TARGET=1`，这将允许插件保持可用 IP 地址的“完全弹性网络接口”。考虑在断开连接状态下根据您的扩缩需求更改 `WARM_ENI_TARGET`、`WARM_IP_TARGET` 和 `MINIMUM_IP_TARGET` 值。有关更多信息，请参阅 GitHub 上插件的 `readme` 文件。有关每种实例类型支持的 Pods 最大数量列表，请参阅 GitHub 上的 `eni-max-pods.txt` 文件。

在网络断开连接期间对您的本地集群进行身份验证

AWS Identity and Access Management（IAM）在网络连接断开期间不可用。在断开连接时，您将无法使用 IAM 凭证对您的本地集群进行身份验证。但是，在断开连接时，您可以使用 `x509` 证书通过本地网络连接到集群。您需要下载并存储客户端 `X509` 证书，以便在断开连接期间使用。在本主题中，您将学习在集群处于断开连接状态时，如何创建和使用证书对其进行身份验证。

1. 创建证书签名请求。

a. 生成证书签名请求。

```
openssl req -new -newkey rsa:4096 -nodes -days 365 \
    -keyout admin.key -out admin.csr -subj "/CN=admin"
```

b. 在 Kubernetes 中创建证书签名请求。

```
BASE64_CSR=$(cat admin.csr | base64 -w 0)
cat << EOF > admin-csr.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. 使用 kubectl 创建证书签名请求。

```
kubectl create -f admin-csr.yaml
```

3. 检查证书签名请求的状态。

```
kubectl get csr admin-csr
```

示例输出如下。

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

Kubernetes 已创建证书签名请求。

4. 批准证书签名请求。

```
kubectl certificate approve admin-csr
```

5. 重新检查证书签名请求状态是否为已批准。

```
kubectl get csr admin-csr
```

示例输出如下。

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Approved
```

6. 检索并验证证书。

   a. 检索证书。

   ```
   kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode
    > admin.crt
   ```

   b. 验证证书。

   ```
   cat admin.crt
   ```

7. 为 admin 用户创建集群角色绑定。

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. 为断开连接状态生成用户范围的 kubeconfig。

   您可以使用已下载的 admin 证书生成 kubeconfig 文件。替换以下命令中的 *my-cluster* 和 *apiserver-endpoint*。

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
```

```
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. 查看您的 kubeconfig 文件。

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. 如果您的 Outpost 上已有投入生产中的服务，请跳过此步骤。如果 Amazon EKS 是在您的 Outpost 上运行的唯一服务，并且 Outpost 当前未投入生产，则您可以模拟网络断开连接。在使用本地集群投入生产之前，请模拟断开连接，以确保在集群处于断开连接状态时可以访问该集群。

    a. 在将您的 Outpost 连接到 AWS 区域 的联网设备上应用防火墙规则。这将断开 Outpost 的服务链接。您无法创建任何新实例。当前正在运行的实例将断开与 AWS 区域 和互联网的连接。

    b. 您可以使用 x509 证书在断开连接时测试与您的本地集群的连接。确保将 kubeconfig 更改为您在上一步创建的 admin.kubeconfig。将 *my-cluster* 替换为您的本地集群的名称。

    ```
    kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
    ```

    如果您在本地集群处于断开连接状态时发现其存在任何问题，则我们建议开具支持票证。

容量注意事项

本主题提供以下指南：为 Outpost 上的本地 Amazon EKS 集群选择 Kubernetes 控制面板实例类型以及（可选）使用置放群组满足高可用性要求。

在 Outpost 上选择要用于您的本地集群的 Kubernetes 控制面板的实例类型（如 m5、c5 或 r5）前，请首先确认可用于 Outpost 配置的实例类型。确定可用实例类型后，请根据您的工作负载所需的节点数量选择实例大小（如 large、xlarge 或 2xlarge）。下表提供了有关选择实例大小的建议。

> **Note**
>
> 必须在 Outpost 上插入实例大小。请确保本地集群生命周期内，有足够的容量可供该大小的三个实例在 Outpost 上运行。有关可用 Amazon EC2 实例类型的列表，请参阅 AWS Outposts 机架功能中的“计算和存储”章节。

节点数量	Kubernetes 控制面板实例实例大小
1–20	large
21–100	xlarge
101–250	2xlarge
251–500	4xlarge

Kubernetes 控制面板的存储需要每个本地集群拥有 246GB 的 Amazon EBS 存储才能满足 etcd 所需的 IOPS。创建本地集群后，系统将为您自动预置 Amazon EBS 卷。

控制面板置放

如果您不指定具有 OutpostConfig.ControlPlanePlacement.GroupName 属性的置放群组，则在您的 Outpost 上可用的基础容量下，为您的 Kubernetes 控制面板预置的 Amazon EC2 实例不会接受任何特定的硬件置放强制执行。

您可以使用置放群组满足 Outpost 上的本地 Amazon EKS 集群的高可用性要求。通过在集群创建期间指定置放群组，您可以影响 Kubernetes 控制面板实例的置放。这些实例分布在独立的基础硬件（机架或主机）上，从而最大限度地减少相关实例对硬件故障事件的影响。

要求

您可以配置的分布类型取决于部署中的 Outpost 机架数量。

- 在单个逻辑 Outpost 中使用一个或两个物理机架的部署：您必须至少具有三台主机，这些主机通过为 Kubernetes 控制面板实例选择的实例类型进行配置。使用主机级分布的分布置放群组可确保所有 Kubernetes 控制面板实例在 Outpost 部署中可用的基础机架内的不同主机上运行。
- 在单个逻辑 Outpost 中使用三个或更多物理机架的部署：您必须至少具有三台主机，这些主机通过为 Kubernetes 控制面板实例选择的实例类型进行配置。使用机架级分布的分布置放群组可确保所有 Kubernetes 控制面板实例在 Outpost 部署中的不同机架上运行。或者，您可以使用上一个选项中描述的主机级分布置放群组。

您负责创建所需的置放群组。您在调用 CreateCluster API 时指定置放群组。有关置放群组及如何创建置放群组的更多信息，请参阅《Amazon EC2 用户指南》中的置放群组。

注意事项

- 指定置放群组后，Outpost 上必须有可用的插槽容量才能成功创建本地 Amazon EKS 集群。容量因您使用的主机或机架分布类型而异。如果没有足够的容量，则集群仍然处于 `Creating` 状态。您可以检查 `DescribeCluster` API 响应运行状况字段上的 `Insufficient Capacity Error`。您必须释放容量才能使创建过程继续进行。
- 在 Amazon EKS 本地集群平台和版本更新期间，使用滚动更新策略将您集群中的 Kubernetes 控制面板实例替换为新实例。在此替换过程中，每个控制面板实例都将终止，从而释放其相应插槽。新的更新实例将预置到相应插槽位置。更新的实例可能会放置在已发布的插槽中。如果该插槽被另一个不相关的实例占用，并且没有符合所需分布拓扑要求的剩余容量，则集群将保持 `Updating` 状态。您可以查看 `DescribeCluster` API 响应运行状况字段上的相应 `Insufficient Capacity Error`。您必须释放容量，以便更新过程继续进行并重新建立之前的高可用性级别。
- 在每个 AWS 区域中，最多可以为每个账户创建 500 个置放群组。有关更多信息，请参阅《Amazon EC2 用户指南》中的一般规则和限制。

对 AWS Outposts 上的 Amazon EKS 的本地集群进行故障排除

本主题将介绍您在使用本地集群时可能会看到的一些常见错误，以及如何排查这些错误。虽然本地集群与云端 Amazon EKS 集群相似，但在 Amazon EKS 的托管方式上存在一些差异。

API 行为

本地集群是通过 Amazon EKS API 创建的，但以异步方式运行。这意味着对于本地集群，对 Amazon EKS API 的请求会立即返回。但是，这些请求可能会成功，也可能由于输入验证错误而快速失效，或者失败并出现描述性验证错误。此行为类似于 Kubernetes API。

本地集群不会过渡到 `FAILED` 状态。Amazon EKS 会尝试以连续方式协调集群状态与用户请求的所需状态。因此，本地集群可能会长时间保持在 `CREATING` 状态，直到根本问题得到解决。

描述集群运行状况字段

可以使用 `describe-cluster` Amazon EKS AWS CLI 命令发现本地集群问题。本地集群问题将由 `describe-cluster` 命令响应的 `cluster.health` 字段显示。此字段中包含的消息包括错误代码、描述性消息和相关的资源 ID。此信息将仅通过 Amazon EKS API 和 AWS CLI 提供。在以下示例中，将 *`my-cluster`* 替换为您的本地集群的名称。

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

示例输出如下。

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

如果问题无法修复，您可能需要删除该本地集群，然后创建一个新的本地集群。例如，尝试使用 Outpost 上不可用的实例类型预调配集群。下表包括常见的运行状况相关错误。

错误情形	代码	消息	ResourceIds
无法找到提供的子网。	ResourceNotFound	The subnet ID *subnet-id* does not exist	所有提供的子网 ID
提供的子网不属于同一 VPC。	ConfigurationConflict	Subnets specified must belong to the same VPC	所有提供的子网 ID
提供的某些子网不属于指定的 Outpost。	ConfigurationConflict	Subnet *subnet-id* expected to be in *outpost-arn*, but is in *other-outpost-arn*	子网 ID 有问题
提供的某些子网不属于任何 Outpost。	ConfigurationConflict	Subnet *subnet-id* is not part of any Outpost	子网 ID 有问题
提供的某些子网没有足够的空闲地址来为	ResourceLimitExceeded	The specified subnet does	子网 ID 有问题

错误情形	代码	消息	ResourceIds
控制面板实例创建弹性网络接口。		`not have enough free addresses to satisfy the request.`	
Outpost 上不支持指定的控制面板实例类型。	`ConfigurationConflict`	`The instance type type is not supported in Outpost outpost-arn`	集群 ARN
您已终止控制面板 Amazon EC2 实例，或 `run-instance` 已成功但观察到的状态更改为 `Terminated` 。在您的 Outpost 重新连接并且 Amazon EBS 内部错误导致 Amazon EC2 内部工作流失败后，这种情况可能会发生一段时间。	`InternalFailure`	`EC2 instance state "Terminated" is unexpected`	集群 ARN
Outpost 容量不足。在集群创建期间，如果 Outpost 与 AWS 区域断开连接，也会发生这种情况。	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	集群 ARN
您的账户已超出安全组配额。	`ResourceLimitExceeded`	Amazon EC2 API 返回的错误消息	目标 VPC ID
您的账户已超出弹性网络接口配额。	`ResourceLimitExceeded`	Amazon EC2 API 返回的错误消息	目标子网 ID

错误情形	代码	消息	ResourceIds
无法通过 AWS Systems Manager 访问控制面板实例。有关分辨率的信息，请参阅 无法通过 AWS Systems Manager 访问控制面板实例。	`ClusterUnreachable`	无法通过 SSM 访问 Amazon EKS 控制面板实例。请验证您的 SSM 和网络配置，并参考 Outpost 上的 EKS 故障排除文档。	Amazon EC2 实例 ID
获取托管安全组或弹性网络接口的详细信息时出错。	基于 Amazon EC2 客户端错误代码。	Amazon EC2 API 返回的错误消息	所有托管安全组 ID
授权或撤销安全组传入规则时出错。这既适用于集群安全组，也适用于控制面板安全组。	基于 Amazon EC2 客户端错误代码。	Amazon EC2 API 返回的错误消息	安全组 ID 有问题
删除控制面板实例的弹性网络接口时出错。	基于 Amazon EC2 客户端错误代码。	Amazon EC2 API 返回的错误消息	弹性网络接口 ID 有问题

下表列出了来自 `describe-cluster` 响应的运行状况字段中呈现的其他 AWS 服务 的错误。

Amazon EC2 错误代码	集群运行状况问题代码	描述
`AuthFailure`	`AccessDenied`	有多种原因可能导致此错误。最常见的原因是，您从控制面板中意外删除了服务用于缩小服务相关角色策略范围的标签。如果出现这种情况，Amazon EKS 将无法继续管理和监控这些 AWS 资源。

Amazon EC2 错误代码	集群运行状况问题代码	描述
UnauthorizedOperation	AccessDenied	有多种原因可能导致此错误。最常见的原因是，您从控制面板中意外删除了服务用于缩小服务相关角色策略范围的标签。如果出现这种情况，Amazon EKS 将无法继续管理和监控这些 AWS 资源。
InvalidSubnetID.NotFound	ResourceNotFound	如果无法找到安全组传入规则的子网 ID，将发生此错误。
InvalidPermission.NotFound	ResourceNotFound	如果安全组传入规则的权限不正确，将发生此错误。
InvalidGroup.NotFound	ResourceNotFound	如果无法找到安全组的传入规则组，将发生此错误。
InvalidNetworkInterfaceID.NotFound	ResourceNotFound	如果无法找到安全组传入规则的网络接口 ID，将发生此错误。
InsufficientFreeAddressesInSubnet	ResourceLimitExceeded	如果超出子网资源配额，将发生此错误。
InsufficientCapacityOnOutpost	ResourceLimitExceeded	如果超出 Outpost 容量配额，将发生此错误。
NetworkInterfaceLimitExceeded	ResourceLimitExceeded	如果超出弹性网络接口配额，将发生此错误。
SecurityGroupLimitExceeded	ResourceLimitExceeded	如果超出安全组配额，将发生此错误。

Amazon EC2 错误代码	集群运行状况问题代码	描述
VcpuLimitExceeded	ResourceLimitExceeded	在新账户中创建 Amazon EC2 实例时，将观察到此情况。错误可能类似于以下内容："You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."
InvalidParameterValue	ConfigurationConflict	如果 Outpost 上不支持指定的实例类型，Amazon EC2 将返回此错误代码。
所有其他故障	InternalFailure	无

无法创建或修改集群

本地集群所需的权限和策略与云端托管的 Amazon EKS 集群需要的权限和策略不同。当集群创建失败并显示 InvalidPermissions 错误时，请仔细检查您使用的集群角色是否附加了 AmazonEKSLocalOutpostClusterPolicy 托管策略。所有其他 API 调用都需要与云中的 Amazon EKS 集群相同的权限集。

集群卡在 **CREATING** 状态

创建本地集群所需的时间取决于多个因素。这些因素包括您的网络配置、Outpost 配置和集群配置。一般情况下，可在 15–20 分钟内创建本地集群并更改为 ACTIVE 状态。如果本地集群仍处于 CREATING 状态，您可以在 cluster.health 输出字段中调用 describe-cluster 以了解关于原因的信息。

最常见的问题如下：

AWS Systems Manager（Systems Manager）会遇到以下问题：

- 您的集群无法从 Systems Manager 所在的 AWS 区域 连接到控制面板实例。您可以通过从区域内的堡垒主机调用 `aws ssm start-session --target instance-id` 来对此进行验证。如果该命令不起作用，请检查 Systems Manager 是否在控制面板实例上运行。或者采用另一种解决方法，即删除集群，然后重新创建集群。
- Systems Manager 控制面板实例可能无法访问互联网。检查您在创建集群时提供的子网是否具有 NAT 网关和带有互联网网关的 VPC。使用 VPC Reachability Analyzer 验证控制面板实例是否可以访问互联网网关。有关更多信息，请参阅 VPC Reachability Analyzer 入门。
- 您提供的角色 ARN 缺少策略。检查是否已从角色中删除 AWS 托管策略：AmazonEKSLocalOutpostClusterPolicy。如果 AWS CloudFormation 堆栈配置错误，也可能会出现这种情况。

在创建集群时错误配置并指定了多个子网：

- 提供的所有子网都必须与同一 Outpost 关联，并且能够相互访问。如果在创建集群期间指定了多个子网，Amazon EKS 会尝试将控制面板实例分布到多个子网中。
- 在弹性网络接口上应用 Amazon EKS 托管安全组。但是，NACL 防火墙规则等其他配置元素也可能会与弹性网络接口的规则冲突。

VPC 和子网 DNS 配置配置错误或缺失

审核Amazon EKS 本地集群 VPC 及子网的要求和注意事项。

无法将节点加入集群

常见原因：

- AMI 问题：
  - 您使用的是不受支持的 AMI。必须使用 v20220620 或更高版本的 Amazon EKS 优化版 Amazon Linux AMI Amazon EKS 优化版 Amazon Linux。
  - 如果您使用 AWS CloudFormation 模板创建您的节点，请确保它没有使用不受支持的 AMI。
- 缺失 AWS IAM 身份验证器 `ConfigMap` – 如果缺失，则必须进行创建。有关更多信息，请参阅 将 `aws-authConfigMap` 应用到集群。

- 使用了错误的安全组 – 请确保将 `eks-cluster-sg-cluster-name-uniqueid` 用于 Worker 节点的安全组。将通过 AWS CloudFormation 更改所选的安全组，以便在每次使用堆栈时允许新的安全组。
- 遵循意外的私有链接 VPC 步骤 - 传递错误的 CA 数据 (`--b64-cluster-ca`) 或 API 端点 (`--apiserver-endpoint`)。
- Pod 安全策略配置错误：
  - CoreDNS 和 Amazon VPC CNI plugin for Kubernetes 守护程序集必须在节点上运行，节点才能加入集群并与之通信。
  - Amazon VPC CNI plugin for Kubernetes 需要一些特权联网功能才能正常运行。您可以使用以下命令查看特权联网功能：`kubectl describe psp eks.privileged`。

  我们建议您不要修改默认容器组（pod）安全策略。有关更多信息，请参阅容器组（pod）安全策略。

收集日志

如果 Outpost 与其关联的 AWS 区域 断开连接，Kubernetes 集群可能会继续正常运行。但是，如果集群无法正常运行，请执行 为网络断开连接做好准备 中的故障排除步骤。如果您遇到其他问题，请联系 AWS Support，AWS Support 会指导您如何下载和运行日志收集工具。这样一来，您就可以从 Kubernetes 集群控制面板实例中收集日志，并将其发送到 AWS Support 支持部门，以便进一步调查。

无法通过 AWS Systems Manager 访问控制面板实例

当无法通过 AWS Systems Manager (Systems Manager) 访问 Amazon EKS 控制面板实例时，Amazon EKS 将为您的集群显示以下错误。

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

要解决此问题，请确保您的 VPC 和子网满足 Amazon EKS 本地集群 VPC 及子网的要求和注意事项 中的要求，并且您已完成《AWS Systems Manager 用户指南》内设置 Session Manager 中的步骤。

在 Outpost 上启动自行管理的 Amazon Linux 节点

本主题介绍如何在向您的 Amazon EKS 集群的 Outpost 上启动 Amazon Linux 节点的自动扩缩组。集群可以位于 AWS Cloud 上，也可以位于 Outpost 上。

先决条件

- 一个现有的 Outpost。有关更多信息，请参阅什么是 AWS Outposts。
- 现有 Amazon EKS 集群。要在 AWS Cloud 上部署集群，请参阅 创建 Amazon EKS 集群。要在 Outpost 上部署集群，请参阅 AWS Outposts 上的 Amazon EKS 的本地集群。
- 假设您正在 AWS Cloud 上的集群中创建节点，并在已启用 AWS Outposts、AWS Wavelength 或 AWS Local Zones 的 AWS 区域 中拥有子网。接下来，当您创建集群时，不得传入这些子网。如果您要在 Outpost 上的集群中创建节点，则在创建集群时必须已传入 Outpost 子网。
- （建议用于 AWS Cloud 上的集群）– 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 策略。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（IRSA）。本地集群不支持服务账户的 IAM 角色。

您可以使用 `eksctl` 或 AWS Management Console（带有 AWS CloudFormation 模板）创建自行管理的 Amazon Linux 节点组。您还可以使用 Terraform。

eksctl

先决条件

您的设备或 AWS CloudShell 上安装了 `0.183.0` 版或更高版本的 `eksctl` 命令行工具。要安装或更新 `eksctl`，请参阅 `eksctl` 文档中的 Installation。

要使用 **`eksctl`** 启动自行管理的 Linux 节点

1. 如果您的集群位于 AWS Cloud 上，并且 AmazonEKS_CNI_Policy 托管 IAM 策略附加到您的 Amazon EKS 节点 IAM 角色，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（IRSA）。如果您的集群位于 Outpost 上，则必须将该策略附加到您的节点角色。
2. 以下命令在现有集群中创建节点组。必须已经使用 `eksctl` 创建了集群。将 *`al-nodes`* 替换为您的节点组名称。节点组名称的长度不能超过 63 个字符。它必须以字母或数字开头，但也可以包括其余字符的连字符和下划线。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。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。如果您的集群存在于 Outpost 上，请将 *`id`* 替换为 Outpost 子网的 ID。如果您的集群存在于 AWS Cloud 上，请将 *`id`* 替换为您在创建集群时未指定的子网的 ID。将 *`instance-type`* 替换为您的 Outpost 支持的实例类型。将剩余的 *`example values`* 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。

将 *instance-type* 替换为您的 Outpost 上可用的实例类型。

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 AWS Management Console 中创建一个。有关更多信息，请参阅《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。

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-
type instance-type \
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2
 --subnet-ids subnet-id
```

如果您的集群部署在 AWS Cloud 上：

- 您部署的节点组可以将 IPv4 地址分配给来自不同 CIDR 块的 Pods，而非实例的。有关更多信息，请参阅 容器组（pod）的自定义网络。
- 您部署的节点组不需要出站互联网访问权限。有关更多信息，请参阅 私有集群要求。

有关所有可用选项和默认设置的完整列表，请参阅 eksctl 文档中的 AWS Outposts 支持。

如果节点无法加入集群，请参阅 Amazon EKS 故障排除 中的 节点未能加入集群 和 对 AWS Outposts 上的 Amazon EKS 的本地集群进行故障排除 中的 无法将节点加入集群。

示例输出如下。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）部署示例应用程序以测试集群和 Linux 节点。

AWS Management Console

步骤 1：使用 AWS Management Console 启动自行管理的 Amazon Linux 节点

1. 下载最新版本的 AWS CloudFormation 模板。

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. 打开 AWS CloudFormation 控制台，地址：https://console.aws.amazon.com/cloudformation。
3. 选择 Create stack（创建堆栈），然后选择 With new resources (standard)（使用新资源（标准））。
4. 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）。选择您在上一步中下载的 `amazon-eks-nodegroup.yaml` 文件，然后选择 Next（下一步）。
5. 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 AWS CloudFormation 堆栈选择堆栈名称。例如，您可以将其命名为 ***al-nodes***。名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母数字字符开头，且不得超过 100 个字符。对于您在其中创建集群的 AWS 区域 和 AWS 账户，该名称必须在其内具有唯一性。
   - ClusterName：输入您的集群的名称。如果此名称与您的集群名称不匹配，则您的节点将无法加入该集群。
   - ClusterControlPlaneSecurityGroup：从您在创建 VPC 时生成的 AWS 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。如果您的集群在 AWS Cloud 上运行，则有关更多信息，请参阅 选择 Amazon EC2 实例类型。如果您的集群在 Outpost 上运行，则您只能选择您的 Outpost 上可用的实例类型。
- NodeImageIdSSMParam：使用用于某个变量 Kubernetes 版本最近的 Amazon EKS 优化版 AMI 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.XX* 替换为不同的支持版本。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。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 AWS 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）。
- NodeVolumeType：指定您的节点的根卷类型。
- 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 AWS Management Console 中创建一个。有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。

  > **Note**
  >
  > 如果此处不提供密钥对，AWS CloudFormation 堆栈创建将失败。

- BootstrapArguments：有几个可选参数可以传递给您的节点。有关更多信息，请查看 GitHub 上的引导脚本使用信息。如果要将节点添加到 AWS Outposts 上的 Amazon EKS 本地集群（其中 Kubernetes 控制面板实例在 AWS Outposts 上运行），并且该集群没有传入和传出互联网连接（也称为私有集群），则必须提供以下引导实际参数（以单行形式）。

```
--b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://
${APISERVER_ENDPOINT} --enable-local-outpost true --cluster-id ${CLUSTER_ID}
```

- 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 IMDSv1，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 的 ID。在选择 VPC 之前，请查看 VPC 要求和注意事项。
- 子网：如果您的集群位于 Outpost 上，则请在您的 VPC 中选择至少一个私有子网。选择子网之前，请查看子网要求和注意事项。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。

6. 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）。
7. 选择 I acknowledge that AWS 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）。
8. 完成创建堆栈后，在控制台中选中它，然后选择 Outputs（输出）。
9. 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

1. 检查您是否已经应用拥有 `aws-auth` `ConfigMap`。

```
kubectl describe configmap -n kube-system aws-auth
```

2. 如果您看到的是 `aws-auth` `ConfigMap`，则请根据需要对其进行更新。

   a. 打开 `ConfigMap` 文件进行编辑。

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 根据需要添加新的 `mapRoles` 条目。将 `rolearn` 值设置为您在上一个步骤中记录的 NodeInstanceRole 值。

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
```

```
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. 保存文件并退出文本编辑器。

3. 如果您收到错误提示 "Error from server (NotFound): configmaps "aws-auth" not found，则请使用库存 ConfigMap。

a. 下载配置映射。

```
curl -O https://s3.us-west-2.amazonaws.com/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 NodeInstanceRole 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
```

4. 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。

> **Note**
>
> 如果您收到任何授权或资源类型错误，请参阅故障排除主题中的 未经授权或访问被拒绝 (kubectl)。

如果节点无法加入集群，请参阅 Amazon EKS 故障排除 中的 节点未能加入集群 和 对 AWS Outposts 上的 Amazon EKS 的本地集群进行故障排除 中的 无法将节点加入集群。

5. 安装 Amazon EBS CSI 驱动程序。有关更多信息，请参阅 GitHub 上的安装。在设置驱动程序权限部分中，确保按照 Using IAM instance profile（使用 IAM 实例配置文件）选项的说明进行操作。您必须使用 gp2 存储类。不支持 gp3 存储类。

   要在您的集群上创建 gp2 存储类，请完成以下步骤。

   1. 运行以下命令以创建 gp2-storage-class.yaml 文件。

      ```
      cat >gp2-storage-class.yaml <<EOF
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        annotations:
          storageclass.kubernetes.io/is-default-class: "true"
        name: ebs-sc
      provisioner: ebs.csi.aws.com
      volumeBindingMode: WaitForFirstConsumer
      parameters:
        type: gp2
        encrypted: "true"
      allowVolumeExpansion: true
      EOF
      ```

   2. 将清单应用于集群。

      ```
      kubectl apply -f gp2-storage-class.yaml
      ```

6. （仅限 GPU 节点）如果选择 GPU 实例类型和 Amazon EKS 优化加速型 AMI，则必须在集群上将 适用于 Kubernetes 的 NVIDIA 设备插件 用作 DaemonSet。将 *vX.X.X* 替换为您需要的 NVIDIA/k8s-device-plugin 版本，然后运行以下命令。

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

第 3 步：其他操作

1. （可选）部署示例应用程序以测试集群和 Linux 节点。
2. 如果您的集群部署在 Outpost 上，则请跳过此步骤。如果您的集群部署在 AWS Cloud 上，以下信息是可选的。如果 AmazonEKS_CNI_Policy 托管 IAM 策略附加到您的 Amazon EKS 节点 IAM 角色，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（IRSA）。

相关项目

这些开源项目扩展了在 AWS 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
- AWS开源博客：eksctl：使用一个命令的 Amazon EKS 集群

AWS Controllers for Kubernetes

使用 AWS Controllers for Kubernetes，您可以直接从 Kubernetes 集群创建和管理 AWS 资源。

- 项目 URL
- AWS 开源博客：Kubernetes 的 AWS 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。有关运算符的更多信息，请参阅 GitHub 上的 OperatorHub.io。

- 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
- AWS 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

适用于 Kubernetes 的 AWS Load Balancer Controller

AWS Load Balancer Controller 有助于管理适用于 Kubernetes 集群的 AWS 弹性负载均衡器。它通过预置 AWS Application Load Balancer 来满足 Kubernetes 入口资源。它通过预置 AWS 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 AWS 服务发现。

- 项目 URL
- 项目文档

机器学习

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
- AWS 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://www.eksworkshop.com/

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AWS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发送给我们的反馈意见。

变更	说明	日期
Kubernetes 版本 1.30	增加了新集群和版本升级的 Kubernetes 版本 1.30 支持。	2024 年 5 月 23 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 Kubernetes 1.29.4、1.28.9 和 1.27.13 的新补丁版本。	2024 年 5 月 14 日
CoreDNS 自动扩缩	CoreDNS 自动扩缩器将根据节点数量和 CPU 核心数量动态调整 EKS 集群中 CoreDNS 部署的副本数量。此功能适用于 CoreDNS v1.9 以及 EKS 发行版 1.25 及更高版本的最新平台版本。	2024 年 5 月 14 日
Windows 的 CloudWatch Container Insights 支持	Amazon CloudWatch Observability Operator 插件现在还允许集群中的 Windows Worker 节点上的 Container Insights。	2024 年 4 月 10 日
Kubernetes 概述	添加了新的 Kubernetes 概念主题。	2024 年 4 月 5 日
重构访问权限和 IAM 内容	将与访问权限和 IAM 主题相关的现有页面（例如身份验证配置映射、访问条目、容器组	2024 年 4 月 2 日

	（pod）ID 和 IRSA）移至新部分。修订概述内容。	
Bottlerocket Amazon S3 CSI 驱动程序的操作系统支持	适用于 Amazon S3 CSI 驱动程序的 Mountpoint 现在与 Bottlerocket 兼容。	2024 年 3 月 13 日
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2024 年 3 月 4 日
Amazon Linux 2023	Amazon Linux 2023（AL2023）是一款新的基于 Linux 的操作系统，旨在为您的云应用程序提供安全、稳定和高性能的环境。	2024 年 2 月 29 日
EKS 容器组身份和 IRSA 支持 Kubernetes1.29 中的附加容器	在 Kubernetes 1.29 中，Amazon EKS 集群中提供附加容器。服务帐户或 EKS 容器组身份的 IAM 角色不支持附加容器。有关附加的更多信息，请参阅 Kubernetes 文档中的附加容器。	2024 年 2 月 26 日
Kubernetes 版本 1.29	增加了新集群和版本升级的 Kubernetes 版本 1.29 支持。	2024 年 1 月 23 日
完整版：Kubernetes版本的 Amazon EKS 延期支持	延期 Kubernetes 版本支持允许您在特定 Kubernetes 版本上停留超过 14 个月。	2024 年 1 月 16 日

AWS Cloud 中的 Amazon EKS 集群运行状况检测	Amazon EKS 会检测到您的 Amazon EKS 集群存在的问题，以及集群运行状况中集群先决条件的基础设施存在的问题。您可以在 AWS Management Console 中以及 EKS API 中集群的 `health` 中查看您的 EKS 集群的问题。除了控制台检测到并显示的问题之外，还有这些问题。以前，集群运行状况仅适用于 AWS Outposts 上的本地集群。	2023 年 12 月 28 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在加拿大西部（卡尔加里）(`ca-west-1`) AWS 区域 推出。	2023 年 12 月 20 日
集群见解	现在，您可以根据定期检查获得有关集群的建议。	2023 年 12 月 20 日
现在，您可以使用访问条目授予 IAM 角色和用户访问您的集群的权限	在引入访问条目之前，您通过向 `aws-auth ConfigMap` 中添加条目来授予 IAM 角色和用户访问您的集群的权限。现在，每个集群都有访问模式，您可以切换到按计划使用访问条目。切换模式后，您可以通过在 AWS CLI、AWS CloudFormation 和 AWS SDK 中添加访问条目来添加用户。	2023 年 12 月 18 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 Kubernetes `1.28.4`、`1.27.8`、`1.26.11` 和 `1.25.16` 的新补丁版本。	2023 年 12 月 12 日

适用于 Amazon S3 的 Mountpoint CSI 驱动程序	现在，您可以在 Amazon EKS 集群上安装适用于 Amazon S3 的 Mountpoint CSI 驱动程序。	2023 年 11 月 27 日
创建集群时开启 Prometheus 指标	在 AWS Management Console 中，您现在可以在创建集群时开启 Prometheus 指标。您还可以在可观测性选项卡中查看 Prometheus 抓取程序详细信息。	2023 年 11 月 26 日
Amazon EKS 容器组身份	Amazon EKS 容器组身份将 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 AWS API。与服务账户的 IAM 角色不同，EKS 容器组身份完全在 EKS 内部；不需要 OIDC 身份提供者。	2023 年 11 月 26 日
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2023 年 11 月 26 日
CSI 快照控制器	现在，您可以安装 CSI 快照控制器，以便与兼容的 CSI 驱动程序一起使用，例如 Amazon EBS CSI 驱动程序。	2023 年 11 月 17 日
ADOT Operator 主题重写	在适用于 OpenTelemetry 的 AWS Distro 文档中，适用于 ADOT Operator 的 Amazon EKS 插件支持部分是多余的。我们将剩余的重要信息迁移到该资源，以减少过时和不一致的信息。	2023 年 11 月 14 日

适用于 Prometheus 指标的 CoreDNS EKS 插件支持	适用于 CoreDNS 的 EKS 插件版本 `v1.10.1-eksbuild.5` 、`v1.9.3-eksbuild.9` 和 `v1.8.7-eksbuild.8` 在 `kube-dns` 服务中公开了端口，以便 CoreDNS 向其发布指标。这样可以轻松将 CoreDNS 指标包含在监控系统中。	2023 年 11 月 10 日
Amazon EKS CloudWatch Observability Operator 插件	添加了 Amazon EKS CloudWatch Observability Operator 页面。	2023 年 11 月 6 日
美国东部（俄亥俄州）自行管理 P5 实例的容量块	在美国东部（俄亥俄州），您现在可以将容量块用于自我管理 P5 实例。	2023 年 10 月 31 日
集群支持修改子网和安全组	您可以更新集群以更改该集群使用的子网和安全组。您可以从 AWS Management Console、AWS CLI 的最新版本、AWS CloudFormation 以及 `eksctl` 版本 `v0.164.0-rc.0` 或更高版本进行更新。您可能需要执行此操作，为子网提供更多可用 IP 地址，以便成功升级集群版本。	2023 年 10 月 24 日

集群角色和托管节点组角色支持客户管理型 AWS Identity and Access Management 策略	您可以对集群角色使用自定义 IAM 策略，而不是 AmazonEKSClusterPolicy AWS 托管策略。此外，您还可以对托管节点组中的节点角色使用自定义 IAM 策略，而不是 AmazonEKSWorkerNodePolicy AWS 托管策略。这样做是为了创建具有最低权限的策略，以满足严格的合规性要求。	2023 年 10 月 23 日
修复 eksctl 安装链接	在移动页面后修复 eksctl 的安装链接。	2023 年 10 月 6 日
预览版：Kubernetes 版本的 Amazon EKS 延期支持	延期 Kubernetes 版本支持允许您在特定 Kubernetes 版本上停留超过 14 个月。	2023 年 10 月 4 日
删除对 AWS App Mesh 集成的引用	Amazon EKS 与 AWS App Mesh 的集成仅适用于 App Mesh 的现有客户。	2023 年 9 月 29 日
Kubernetes 版本 1.28	增加了新集群和版本升级的 Kubernetes 版本 1.28 支持。	2023 年 9 月 26 日
现有集群支持在 Amazon VPC CNI plugin for Kubernetes 中强制执行 Kubernetes 网络策略	您可以将现有集群中的 Kubernetes 网络策略与 Amazon VPC CNI plugin for Kubernetes 结合使用，而不需要第三方解决方案。	2023 年 9 月 15 日
CoreDNS Amazon EKS 附加组件支持修改 PDB	在版本 v1.9.3-eksbuild.7 及更高版本和 v1.10.1-eksbuild.4 及更高版本中，您可以修改 CoreDNS EKS 附加组件的 PodDisruptionBudget 。	2023 年 9 月 15 日

Amazon EKS 对共享子网的支持	关于在共享子网中创建 Amazon EKS 集群的最新共享子网要求和注意事项。	2023 年 9 月 7 日
更新了“什么是 Amazon EKS？”	新增了常见用例和架构主题。更新了其他主题。	2023 年 9 月 6 日
Amazon VPC CNI plugin for Kubernetes 中的 Kubernetes 网络策略执行	您可以将 Kubernetes 网络策略与 Amazon VPC CNI plugin for Kubernetes 结合使用，而不需要第三方解决方案。	2023 年 8 月 29 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在以色列（特拉维夫）（`il-central-1`）AWS 区域开放。	2023 年 8 月 1 日
适用于 Fargate 的可配置临时存储	您可以增加在 running on Amazon EKS Fargate 上运行的每个 Pod 的临时存储总量。	2023 年 7 月 31 日
对 Amazon EFS CSI 驱动程序的附加组件支持	您现在可以使用 AWS Management Console、AWS CLI 和 API 来管理 Amazon EFS CSI 驱动程序。	2023 年 7 月 26 日
AWS 托管策略更新：新策略	Amazon EKS 添加了新的 AWS 托管策略。	2023 年 7 月 26 日
Kubernetes 的 1.27、1.26、1.25 和 1.24 版本更新现在可用于 AWS Outposts 上的本地集群	Kubernetes 的 1.27.3、1.26.6、1.25.11 和 1.24.15 版本更新现在可用于 AWS Outposts 上的本地集群	2023 年 7 月 20 日
Windows 节点的 IP 前缀支持	为节点分配 IP 前缀可以使您在节点上托管的 Pods 数量比为节点分配单个辅助 IP 地址时多得多。	2023 年 7 月 6 日

适用于 OpenZFS CSI 驱动程序的 Amazon FSx	现在，您可以在 AmazonEKS 集群上安装 Amazon FSx for OpenZFS 驱动程序。	2023 年 6 月 30 日
IPv4 集群中 Linux 节点上的 Pods 现在可以与 IPv6 端点通信。	为您的节点分配 IPv6 地址后，Pods 的 IPv4 地址是转换为其所运行节点 IPv6 地址的网络地址。	2023 年 6 月 19 日
AWS GovCloud (US) Regions 中的 Windows 托管节点组	在 AWS GovCloud (US) Regions 中，Amazon EKS 托管节点组现在可以运行 Windows 容器。	2023 年 5 月 30 日
Kubernetes 版本 1.27	增加了新集群和版本升级的 Kubernetes 版本 1.27 支持。	2023 年 5 月 24 日
Kubernetes 版本 1.26	增加了新集群和版本升级的 Kubernetes 版本 1.26 支持。	2023 年 4 月 11 日
无域 gMSA	您现在可以将无域 gMSA 与 Windows Pods 结合使用。	2023 年 3 月 27 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在亚太地区（墨尔本）(ap-southeast-4) AWS 区域 推出。	2023 年 3 月 10 日
Amazon File Cache CSI 驱动程序	现在，您可以在 AmazonEKS 集群上安装 Amazon File Cache CSI 驱动程序。	2023 年 3 月 3 日
Kubernetes 版本 1.25 现在可用于 AWS Outposts 上的本地集群	现在，您可以使用 Kubernetes 版本 1.22 到 1.25 在 Outpost 上创建 Amazon EKS 本地集群。	2023 年 3 月 1 日
Kubernetes 版本 1.25	增加了新集群和版本升级的 Kubernetes 版本 1.25 支持。	2023 年 2 月 22 日

AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2023 年 2 月 7 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在亚太地区（海得拉巴）（ap-south-2 ）、欧洲（苏黎世）（eu-central-2 ）和欧洲（西班牙）（eu-south-2 ）AWS 区域推出。	2023 年 2 月 6 日
Kubernetes 版本 1.21 到 1.24 现在可用于 AWS Outposts 上的本地集群。	现在，您可以使用 Kubernetes 版本 1.21 到 1.24 在 Outpost 上创建 Amazon EKS 本地集群。以前，只有版本 1.21 可用。	2023 年 1 月 17 日
Amazon EKS 现在支持 AWS PrivateLink	您可以使用 AWS PrivateLink 在您的 VPC 和 Amazon EKS 之间创建私有连接。	2022 年 12 月 16 日
托管节点组 Windows 支持	您现在可以将 Windows 用于 Amazon EKS 托管节点组。	2022 年 12 月 15 日
来自独立软件供应商的 Amazon EKS 附加组件现已在 AWS Marketplace 中提供	现在，您可以通过 AWS Marketplace 浏览和订阅独立软件供应商提供的 Amazon EKS 附加组件。	2022 年 11 月 28 日
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2022 年 11 月 17 日
Kubernetes 版本 1.24	增加了新集群和版本升级的 Kubernetes 版本 1.24 支持。	2022 年 11 月 15 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在中东（阿联酋）(me-central-1) AWS 区域 推出。	2022 年 11 月 3 日

AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2022 年 10 月 24 日
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2022 年 10 月 20 日
AWS Outposts 上的本地集群现已推出	现在，您可以在 Outpost 上创建 Amazon EKS 本地集群。	2022 年 9 月 19 日
基于 Fargate vCPU 的配额	Fargate 正在从基于的 Pod 限额过渡到基于 vCPU 的限额。	2022 年 9 月 8 日
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2022 年 8 月 31 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
AWS 托管策略更新：新策略	Amazon EKS 添加了新的 AWS 托管策略。	2022 年 8 月 24 日
AWS 托管策略更新：新策略	Amazon EKS 添加了新的 AWS 托管策略。	2022 年 8 月 23 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 AWS Management Console 中的 Kubernetes 资源	现在，您可以查看使用 AWS Management Console 部署到集群的 Kubernetes 资源的相关信息。	2022 年 5 月 3 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在亚太地区（雅加达）(ap-southeast-3) AWS 区域 推出。	2022 年 5 月 2 日
可观测性页面和 ADOT 插件支持	添加了 Observability（可观察性）页面和 AWS 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
AWS 托管策略更新：新策略	Amazon EKS 添加了新的 AWS 托管策略。	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 AWS Management Console、AWS CLI 和 API来管理 Amazon EBS CSI 驱动程序。	2022 年 3 月 31 日
AWS Outposts 内容更新	在 AWS Outposts 上部署 Amazon EKS 集群的说明。	2022 年 3 月 22 日
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2022 年 3 月 21 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
AWS 托管式策略更新 — 对现有策略的更新	Amazon EKS 更新了现有的 AWS 托管策略。	2021 年 12 月 13 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 AWS Management Console、AWS 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 AWS，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
适用于 NetApp ONTAP CSI 驱动程序的 Amazon FSx	添加了总结适用于 NetApp ONTAP CSI 驱动程序的 Amazon FSx 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加组件软件。	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
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
Kubernetes 版本 1.21	增加了 Kubernetes 版本 1.21 支持。	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 AWS Load Balancer Controller2.2.0	现在，您可以使用 AWS 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 AWS Identity and Access Management (IAM) 一起使用或作为其替代方法。	2021 年 2 月 12 日
在 AWS Management Console 中查看节点和工作负载资源	您现在可以在 AWS 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。	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 AWS 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 AWS Fargate 的 EFS 支持	您现在可以将 Amazon EFS 与 AWS 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 AWS 网络负载均衡器。	2020 年 8 月 12 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在非洲（开普敦）（af-south-1）和欧洲（米兰）（eu-south-1）AWS 区域推出。	2020 年 8 月 6 日
Fargate 使用情况指标	AWS 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	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 AWS 区域 扩展	Amazon EKS 现已在 AWS GovCloud（美国东部）（us-gov-east-1）和 AWS GovCloud（美国西部）（us-gov-west-1）AWS 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	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	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 AWS 区域 扩展	Amazon EKS 现已在北京（cn-north-1）和宁夏（cn-northwest-1）AWS 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 AWS Outpost 上创建集群	Amazon EKS 现在支持在 AWS Outposts 上创建集群。	2019 年 12 月 3 日
Amazon EKS 上的 AWS Fargate	Amazon EKS Kubernetes 集群现在支持在 Fargate 上运行 Pods。	2019 年 12 月 3 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在加拿大（中部）(`ca-central-1`) AWS 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 AWS 区域 扩展	Amazon EKS 现已在南美洲（圣保罗）(`sa-east-1`) AWS 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 Kubernetes1.11	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	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 AWS API。	2019 年 9 月 3 日
Amazon EKS AWS 区域 扩展	Amazon EKS 现已在中东（巴林）(me-south-1) AWS 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	Amazon EKS 现已在亚太地区（香港）(ap-east-1) AWS 区域推出。	2019 年 7 月 31 日
Amazon EKS 不再支持 Kubernetes1.10	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 AWS 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	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 Kubernetes1.10 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
AWS CLIget-token 命令	aws eks get-token 命令已添加到 AWS CLI。您不再需要安装适用于 Kubernetes 的 AWS 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 AWS 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 AWS Command Line Interface 用户指南 中的安装 AWS 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	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 AWS 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 AWS 区域中创建的集群数从 3 增加到 50。	2019 年 2 月 13 日
Amazon EKS AWS 区域 扩展	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` ）AWS 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 AWS 区域 扩展	Amazon EKS 现已在亚太地区（首尔）(`ap-northeast-2`) AWS 区域推出。	2019 年 1 月 9 日

Amazon EKS 区域扩展	Amazon EKS 现已在以下其他AWS 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 AWS 区域 扩展	Amazon EKS 现已在欧洲（斯德哥尔摩）(eu-north-1) AWS 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 AWS 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 AWS CLIupdate-kubeconfig 命令的说明	Amazon EKS 向 AWS 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 AWS 区域 扩展	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 AWS 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	服务的初始文档发布	2018 年 6 月 5 日