aws

用户指南

AWS Batch

AWS Batch: 用户指南

Table of Contents

什么是 AWS Batch？

借助 AWS Batch，您可以在 AWS Cloud上运行批处理计算工作负载。批量计算是开发人员、科学家和工程师用来访问大量计算资源的常见方法。AWS Batch 将会消除配置和管理所需基础设施的千篇一律的繁重工作，与传统批量计算软件相似。此服务可以有效地预配置资源以响应提交的作业，以便消除容量限制、降低计算成本和快速交付结果。

作为一项完全托管服务，AWS Batch 有助于您运行任意规模的批量计算工作负载。AWS Batch 将根据工作负载的数量和规模自动预置计算资源并优化工作负载分配。有了 AWS Batch 之后，不再需要安装或管理批量计算软件，从而使您可以将时间放在分析结果和解决问题上。

主题

AWS Batch 的组成部分

AWS Batch 可让您轻松地在一个区域内跨多个可用区运行批处理任务。您可以在新的或现有的 VPC 中创建 AWS Batch 计算环境。在计算环境就绪并与任务队列关联后，您可以定义任务定义，以指定要运行任务的 Docker 容器映像。容器映像将在容器注册表中存储和提取，可能存在于您的 AWS 基础设施的内部或外部。

任务

提交到 AWS Batch 的工作单位 (如 shell 脚本、Linux 可执行文件或 Docker 容器映像)。它具有名称，并在您的计算环境中的 AWS Fargate 或 Amazon EC2 资源上作为容器化应用程序运行 (使用您在任务定义中指定的参数)。任务可以按名称或按 ID 引用其他任务，并且可以依赖于其他任务的成功完成。有关更多信息，请参阅作业。

作业定义

作业定义指定作业的运行方式。您可以把作业定义看成是任务中的资源的蓝图。您可以为作业提供 IAM 角色以提供对其他 AWS 资源的访问权限。您还可以指定内存和 CPU 要求。任务定义还可以控制容器属性、环境变量和持久性存储的挂载点。任务定义中的许多规范可以通过在提交单个任务时指定新值来覆盖。有关更多信息，请参阅作业定义。

作业队列

当您提交 AWS Batch 任务时，会将其提交到特定的任务队列中，然后它驻留在那里直到被安排到计算环境中为止。将一个或多个计算环境与作业队列相关联。您还可以为这些计算环境甚至作业队列本身分配优先级值。例如，您可以有一个高优先级队列用以提交时间敏感型任务，以及一个低优先级队列供可在计算资源较便宜时随时运行的任务使用。

计算环境

计算环境是一组用于运行任务的托管或非托管计算资源。在托管计算环境中，您可以按多个详细级别指定所需的计算类型（Fargate 或 EC2）。您可以设置使用特定类型 EC2 实例的计算环境，例如 `c5.2xlarge` 或 `m5.10xlarge`。或者，您也可以选择仅指定要使用最新的实例类型。您还可以指定环境的最小、期望和最大 vCPU 数量，以及您愿意为竞价型实例支付的金额占按需型实例价格的百分比以及目标 VPC 子网集。AWS Batch 将根据需要有效地启动、管理和终止计算类型。您还可以管理自己的计算环境。因此，您负责在 AWS Batch 为您创建的 Amazon ECS 集群中设置和扩展实例。有关更多信息，请参阅计算环境。

入门

通过在 AWS Batch 控制台中创建任务定义、计算环境和任务队列来开始使用 AWS Batch。

AWS Batch 首次运行向导为您提供了创建计算环境和作业队列并提交示例 Hello World 作业的选项。如果您具有要在 AWS Batch 中启动的 Docker 映像，则可以使用此映像创建作业定义并改为将此定义提交到您的队列。有关更多信息，请参阅入门 AWS Batch。

控制面板

在 AWS Batch 控制面板上，您可以监控最近的作业、作业队列和计算环境。默认情况下，会显示以下控制面板小组件：

- 作业概述 - 有关 AWS Batch 作业的更多信息，请参阅 作业。
- 作业队列概述 - 有关 AWS Batch 作业队列的更多信息，请参阅 作业队列。
- 计算环境概述 - 有关 AWS Batch 计算环境的更多信息，请参阅 计算环境。

您可以自定义“控制面板”页面上显示的小组件。以下各节描述了您可以安装的其他小组件。

单个作业队列

此小组件显示有关单个作业队列的详细信息。

要添加此小组件，请按照以下步骤操作。

1. 打开 AWS Batch 控制台。
2. 从导航栏中，选择您想要的 AWS 区域。
3. 在导航窗格中，选择 Dashboard (控制面板)。
4. 选择添加小部件。
5. 对于单一作业队列，选择添加小组件。
6. 对于作业队列，选择所需的作业队列。
7. 对于作业状态，选择要显示的作业状态。
8. （可选）如果您不想显示计算环境的属性，请禁用显示连接的计算环境。
9. 在计算环境属性中，选择所需的属性。
10选择 Add（添加）。

CloudWatch Container Insights

此小组件显示 AWS Batch 计算环境和作业的汇总指标。有关安装 Container Insights 的更多信息，请参阅 CloudWatch Container Insights。

要添加此小组件，请按照以下步骤操作。

1. 打开 AWS Batch 控制台。
2. 从导航栏中，选择您想要的 AWS 区域。
3. 在导航窗格中，选择 Dashboard (控制面板)。
4. 选择添加小部件。
5. 对于容器洞察，请选择添加小组件。
6. 对于计算环境，选择所需的计算环境。
7. 选择 Add（添加）。

作业日志

此小组件在一个方便的位置显示作业的不同日志。有关作业日志的更多信息，请参阅 the section called “ 作业日志”。

要添加此小组件，请按照以下步骤操作。

1. 打开 AWS Batch 控制台。
2. 从导航栏中，选择您想要的 AWS 区域。
3. 在导航窗格中，选择 Dashboard (控制面板)。
4. 选择添加小部件。
5. 对于作业日志，选择添加小组件。
6. 在作业 ID 中，输入所需作业的作业 ID。
7. 选择 Add（添加）。

使用进行设置 AWS Batch

如果您已经注册了 Amazon Web Services (AWS)，并且正在使用 Amazon Elastic Compute Cloud (Amazon EC2)，或 Amazon Elastic Container Service (Amazon ECS)，那么您很快就可以使用 AWS Batch。这些服务的设置过程相似。这是因为在其计算环境中 AWS Batch 使用了 Amazon ECS 容器实例。要与 AWS CLI 一起使用 AWS Batch ，必须使用支持最新 AWS Batch 功能 AWS CLI 的版本。如果您在中看不到对某项 AWS Batch 功能的支持 AWS CLI，请升级到最新版本。欲了解更多信息，请参阅 http://aws.amazon.com/cli/。

> **Note**
>
> 由于 AWS Batch 使用 Amazon EC2 的组件，因此您可以使用 Amazon EC2 控制台执行其中的许多步骤。

完成以下任务进行准备 AWS Batch。如果您已完成以下任何步骤，可以直接跳到安装 AWS CLI的步骤。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

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

在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

为您的计算环境和容器实例创建 IAM 角色

您的 AWS Batch 计算环境和容器实例需要 AWS 账户 凭据才能代表您调用其他 AWS API。创建可将这些凭证提供给计算环境和容器实例的 IAM 角色，然后将该角色与计算环境关联。

> **Note**
>
> AWS Batch 计算环境和容器实例角色是在控制台首次运行体验中自动为您创建的。因此，如果您打算使用 AWS Batch 控制台，则可以继续阅读下一部分。如果您打算 AWS CLI 改用，Amazon ECS 实例角色请在创建第一个计算环境之前完成将服务相关角色用于 AWS Batch和中的步骤。

创建密钥对

AWS 使用公钥加密来保护您的实例的登录信息。Linux 实例（例如 AWS Batch 计算环境容器实例）没有可用于 SSH 访问的密码。您使用密钥对安全地登录到实例。您可以在创建计算环境时指定密钥对的名称，然后在使用 SSH 登录时提供私有密钥。

如果您尚未创建密钥对，则可以通过 Amazon EC2 控制台自行创建。请注意，如果您计划启动多个实例 AWS 区域，请在每个区域创建一个 key pair。有关区域的更多信息，请参阅 Amazon EC2 用户指南中的区域和可用区。

创建密钥对

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

2. 在导航栏中，AWS 区域 为密钥对（key pair）选择一个。您可以选择对您可用的任意区域，无论您的位置如何；但是，密钥对是特定于区域的。例如，如果您计划在美国西部（俄勒冈州）中启动实例，则必须在同一区域中为实例创建密钥对。
3. 在导航窗格中，选择 Key Pairs 和 Create Key Pair。
4. 在 Create Key Pair 对话框中，为 Key pair name 输入新密钥对的名称，然后选择 Create。选择一个可以记住的名称，例如您的用户名，后跟 `-key-pair`，并加区域名称。例如，me-key-pair-uswest2。
5. 您的浏览器会自动下载私有密钥文件。基本文件名是您为密钥对指定的名称，文件扩展名为 `.pem`。将私有密钥文件保存在安全位置。

   > **Important**
   >
   > 这是您保存私有密钥文件的唯一机会。启动实例时，您需要提供密钥对的名称；每次连接到实例时，必须提供相应的私有密钥。

6. 如果您将在 Mac 或 Linux 计算机上使用 SSH 客户端连接到您的 Linux 实例，请使用以下命令设置您私有密钥文件的权限。这样，只有你才能读懂。

```
$ chmod 400 your_user_name-key-pair-region_name.pem
```

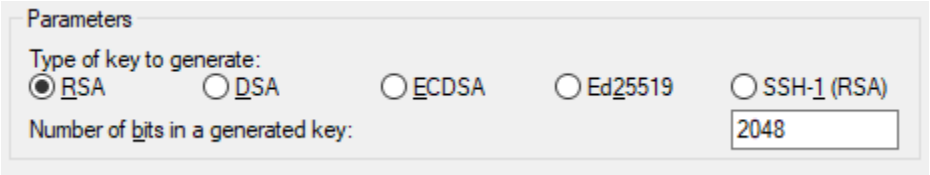
有关更多信息，请参阅《亚马逊 EC2 用户指南》中的 Amazon EC2 密钥对。

使用密钥对连接到实例

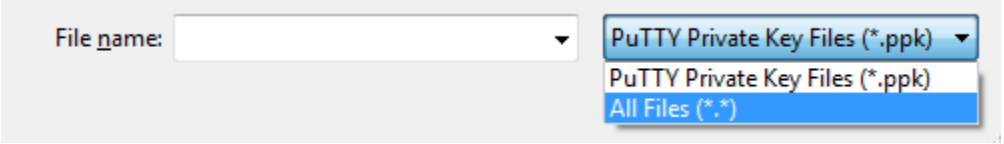
要从运行 Mac 或 Linux 的计算机连接到 Linux 实例，需要使用 `-i` 选项对 SSH 客户端指定 `.pem` 文件和私有密钥的路径。要从运行 Windows 的计算机连接到你的 Linux 实例，请使用 MindTerm 或 Putty。如果您计划使用 PuTTY，请安装它并遵循以下过程将 `.pem` 文件转换为 `.ppk` 文件。

（可选）准备使用 PuTTY 从 Windows 连接到 Linux 实例

1. 从 http://www.chiark.greenend.org.uk/~sgtatham/putty/ 下载并安装 PuTTY。确保安装整个套件。
2. 启动 PuTTYgen（例如，在 开始菜单中，依次选择 所有程序、PuTTY 和 PuTTYgen）。
3. 在 Type of key to generate 下，选择 RSA。如果您使用的是旧版本的 PuTTYgen，请选择 SSH-2 RSA。



4. 选择 Load。默认情况下，PuTTYgen 仅显示扩展名为 .ppk 的文件。要找到您的 .pem 文件，请选择显示所有类型的文件的选项。



5. 选择您在上一个过程中创建的私有密钥文件，然后选择 Open。选择 OK 关闭确认对话框。
6. 选择保存私有密钥。PuTTYgen 显示一条关于在没有密码的情况下保存密钥的警告。选择是。
7. 指定与密钥对相同的密钥名称。PuTTY 会自动添加 .ppk 文件扩展名。

创建 VPC

借助 Amazon Virtual Private Cloud（亚马逊 VPC），您可以将 AWS 资源启动到您定义的虚拟网络中。我们强烈建议您在 VPC 中启动您的容器实例。

如果您有默认 VPC，也可以跳过此部分并进入下一个任务 创建安全组。要确定您是否有默认 VPC，请参阅《亚马逊 EC2 用户指南》中的 Amazon EC2 控制台中支持的平台

有关如何创建 Amazon VPC 的信息，请参阅 Amazon VPC 用户指南中的仅创建 VPC。请参阅下表确定需要选择的选项。

选项	值
要创建的资源	仅限 VPC
名称	可以选择为您的 VPC 提供名称。
IPv4 CIDR 块	IPv4 CIDR 手动输入 CIDR 块大小必须在 /16 和 /28 之间。
IPv6 CIDR 块	无 IPv6 CIDR 块

选项	值	
租赁	默认	

有关 Amazon VPC 的更多信息，请参阅 Amazon VPC 用户指南中的什么是Amazon VPC？。

创建安全组

安全组用作关联的计算环境容器实例的防火墙，可在容器实例级别控制入站和出站的数据流。安全组只能在为其创建该组的 VPC 中使用。

您可以向安全组添加规则，以便使用 SSH 从您的 IP 地址连接到容器实例。您还可以添加允许来自任意位置的入站和出站 HTTP 和 HTTPS 访问的规则。向任务所需的开放端口添加任意规则。

请注意，如果您计划在多个区域中启动容器实例，则需要在每个区域中创建安全组。有关更多信息，请参阅《Amazon EC2 用户指南》中的区域和可用区。

> **Note**
>
> 您需要本地计算机的公有 IP 地址，可以使用服务获得该地址。例如，我们提供以下服务：http://checkip.amazonaws.com/ 或 https://checkip.amazonaws.com/。要查找另一项可提供您的 IP 地址的服务，请使用搜索短语“what is my IP address”。如果您正通过互联网服务提供商（ISP）连接或者在不使用静态 IP 地址的情况下从防火墙后面连接，则找出客户端计算机使用的 IP 地址范围。

使用控制台创建安全组

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航窗格中，选择 Security Groups（安全组）。
3. 选择Create security group（创建安全组）。
4. 输入安全组的名称和描述。在创建安全组后，您无法更改其名称和描述。
5. 从 VPC 中，选择 VPC。
6. （可选）在默认情况下，新安全组起初只有一条出站规则，即允许所有通信离开资源。您必须添加规则，以便允许任何入站数据流或限制出站数据流。

AWS Batch 容器实例不需要打开任何入站端口。但是，您可能需要添加 SSH 规则。这样，您就可以登录容器实例并使用 Docker 命令检查作业中的容器。如果您希望容器实例托管运行 Web 服务器的作业，也可以添加适用于 HTTP 的规则。完成以下步骤可添加这些可选的安全组规则。

在 Inbound 选项卡上，创建以下规则并选择 Create：

- 选择添加规则。对于类型，选择 HTTP。对于 Source，选择 Anywhere (`0.0.0.0/0`)。
- 选择添加规则。对于 Type，选择 SSH。对于源，选择自定义 IP，然后以无类别域间路由 (CIDR) 表示法指定计算机或网络的公有 IP 地址。如果您的公司要分配同一范围内的地址，请指定整个范围，例如 `203.0.113.0/24`。要采用 CIDR 表示法指定单个 IP 地址，请选择我的 IP。这会将路由前缀 `/32` 添加到公有 IP 地址。

> **Note**
>
> 出于安全原因，我们不建议您允许从所有 IP 地址 (`0.0.0.0/0`) 对您的实例进行 SSH 访问，但仅用于测试目的，并且仅在短时间内进行。

7. 您可以现在添加标签，也可以稍后再添加。要添加标签，请选择 Add new tag（添加新标签），然后输入标签键和值。
8. 选择Create security group（创建安全组）。

要使用命令行创建安全组，请参见 create-security-group (AWS CLI)

更多有关安全组的信息，请参阅使用安全组部分。

安装 AWS CLI

要与 AWS CLI 一起使用 AWS Batch，请安装最新 AWS CLI 版本。有关安装 AWS CLI 或将其升级到最新版本的信息，请参阅《AWS Command Line Interface 用户指南》中的安装 AWS 命令行界面。

入门 AWS Batch

您可以使用 AWS Batch 首次运行向导快速入门。 AWS Batch完成先决条件后，您可以使用首次运行向导来创建计算环境、作业定义和作业队列。

您也可以使用 AWS Batch 首次运行向导提交 “Hello World” 作业示例，以测试您的配置。如果您已经有想要启动的 Docker 镜像 AWS Batch，则可以使用该镜像来创建任务定义。

先决条件

在启动 AWS Batch 首次运行向导之前，请务必执行以下操作：

- 完成 使用进行设置 AWS Batch 中所述的步骤。
- 验证您是否 AWS 账户 具有所需的权限。

Amazon EC2 入门

Amazon Elastic Compute Cloud (Amazon EC2) 在 AWS Cloud中提供可扩展的计算容量。使用 Amazon EC2 可避免前期的硬件投入，因此您能够快速开发和部署应用程序。

您可以使用 Amazon EC2 启动所需数量的虚拟服务器，配置安全性和联网以及管理存储。Amazon EC2 可让您扩展或缩减以处理需求变化或使用高峰，从而减少预测流量的需求。

创建计算环境

要为 Amazon EC2 编排创建计算环境，请执行以下操作：

1. 打开 AWS Batch 控制台首次运行向导。
2. 对于选择编排类型，选择 Amazon Elastic Compute Cloud (Amazon EC2)。
3. 选择下一步。
4. 在名称的计算环境配置部分，为您的计算环境指定唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
5. 对于 实例角色，请选择使用附加了所需 IAM 权限的现有实例配置文件。此实例配置文件允许您的计算环境中的 Amazon ECS 容器实例调用所需的 AWS API 操作。有关更多信息，请参阅 Amazon ECS 实例角色。

6. （可选）标签是为资源分配的标记。要添加标签或 Amazon EC2 标签，请展开标签，然后选择添加标签。输入一个键值对，然后再次选择添加标签。

> **Important**
>
> 如果选择添加标签，则必须输入键值对，然后再次选择添加标签或选择移除标签。

7. （可选）在使用 Amazon EC2 竞价型实例的实例配置部分，开启使用竞价型实例启用。
8. （仅限 Spot）对于按需价格的最大百分比，请输入您要为竞价资源支付的最大按需定价百分比。
9. （可选）（仅限 Spot）对于竞价型实例集角色，选择一个现有的 Amazon EC2 竞价型实例集 IAM 角色以应用于您的 Spot 计算环境。如果您没有现有的 Amazon EC2 竞价型实例集 IAM 角色，则必须先创建一个。有关更多信息，请参阅 Amazon EC2 竞价型实例集角色。

> **Important**
>
> 要在创建时标记您的竞价型实例，您的 Amazon EC2 竞价型队列 IAM 角色必须使用更新的 A mazonEC2 SpotFleetTaggingRole 托管策略。A mazonEC2 SpotFleetRole 托管策略没有标记竞价型实例所需的权限。有关更多信息，请参阅 创建时未标记的竞价型实例 和 the section called “对资源加标签”。

10. 对于 Minimum vCPUs，选择您的计算环境应保留的 EC2 vCPU 的最小数目，而无论作业队列需求如何。
11. 对于 Desired vCPUs，请选择您的计算环境在启动时应使用的 EC2 vCPU 数量。随着作业队列需求的增加， AWS Batch 所需的 vCPU 数量增加并添加 EC2 实例。vCPU 的数量可以增加到 vCPU 的最大数量。随着需求的 AWS Batch 减少，减少所需的 vCPU 数量并移除实例。一直减少到 vCPU 的最小数量。
12. 对于 Maximum vCPUs，选择您的计算环境可以横向扩展到的 EC2 vCPU 的最大数目，而无论作业队列需求如何。
13. 对于允许的实例类型，选择可启动的 Amazon EC2 实例类型。您可以指定实例系列以在这些系列中启动任何实例类型（例如，`c5`、`c5n` 或 `p3`），或者，您可以指定系列中的特定大小（例如 `c5.8xlarge`）。Metal 实例类型不在实例系列中。例如，`c5` 不包括 `c5.metal`。还可以通过选择 `optimal` 来选择符合作业队列需求的实例类型（从 C4、M4 和 R4 实例系列中）。

> **Note**
>
> 在创建一个计算环境时，为该计算环境选择的实例类型必须共享同一架构。例如，您不能在同一个计算环境中混用 x86 和 ARM 实例。

> **Note**
>
> AWS Batch 根据任务队列中所需的数量扩展 GPU。要使用 GPU 计划，计算环境必须包含 p2, p3, p4, p5, g3, g3s, g4 或 g5 系列的实例类型。

> **Note**
>
> 目前，`optimal` 使用 C4、M4 和 R4 实例系列中的实例类型。如果没有 AWS 区域 这些实例系列的实例类型，则使用C5M5、和R5实例系列中的实例类型。

14. 展开其他配置。
15. （可选）对于置放群组，输入置放群组名称以对计算环境中的资源进行分组。
16. （可选）对于 EC2 密钥对，请在连接到实例时选择公钥和私有密钥对作为安全凭证。有关 Amazon EC2 密钥对的更多信息，请参阅 Amazon EC2 密钥对和 Linux 实例。
17. 对于分配策略，选择在从允许的实例类型列表中选择实例类型时要使用的分配策略。对于 EC2 按需计算环境，BEST_FIT_PROGRESSIVE 通常是更好的选择，而对于 EC2 Spot 计算环境，SPOT_CAPACITY_OPTIMIZED 则是更好的选择。有关更多信息，请参阅 the section called “分配策略”。
18. （可选）对于 EC2 配置，请选择添加 EC2 配置。选择图像类型和映像 ID 覆盖值以提供信息，AWS Batch 以便为计算环境中的实例选择 Amazon 系统映像 (AMI)。如果未为每种图像类型指定镜像 ID 覆盖，则 AWS Batch 选择最近经过亚马逊ECS优化的 AMI。如果未指定图像类型，则对于非 GPU、非 G AWS raviton 实例，默认为 Amazon Linux 2。

> **Important**
>
> 要使用自定义 AMI，请选择映像类型，然后在映像 ID 覆盖框中输入自定义 AMI ID。

Amazon Linux 2

所有 AWS 基于 Graviton 的实例系列（例如、C6gM6g、和T4g）均为默认值R6g，并且可用于所有非 GPU 实例类型。

Amazon Linux 2（GPU）

所有 GPU 实例系列的默认值（例如P4和G4），并且可用于所有非 AWS 基于 Graviton 的实例类型。

Amazon Linux

可用于非 GPU、非 G AWS raviton 实例系列。对 Amazon Linux AMI 的标准支持已结束。有关更多信息，请参阅 Amazon Linux AMI。

> **Note**
>
> 您为计算环境选择的 AMI 必须与您希望用于该计算环境的实例类型的架构匹配。例如，如果您的计算环境使用 A1 实例类型，则您选择的计算资源 AMI 必须支持 Arm 实例。Amazon ECS 同时提供经过 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 Arm 版本。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的经过 Amazon ECS 优化的 Amazon Linux 2 AMI。

19. （可选）对于启动模板，选择现有的 Amazon EC2 启动模板来配置您的计算资源。模板的默认版本会自动填充。有关更多信息，请参阅 启动模板支持。

> **Note**
>
> 在启动模板中，您可以指定自己创建的自定义 AMI。

20. （可选）对于 Launch template version (启动模板版本)，输入 $Default、$Latest 或要使用的特定版本号。

> **Important**
>
> 创建计算环境后，即使更新启动模板的 $Default 或 $Latest 版本，也不会更改使用的启动模板版本。要使用新的启动模板版本，请先创建新的计算环境，将新的计算环境添加到现有作业队列。然后，从作业队列中移除旧的计算环境，然后删除旧的计算环境。

21. 在网络配置部分：

    a. 对于虚拟私有云（VPC）ID，选择一个 Amazon VPC。

    b. 对于子网，将列出您的 AWS 账户 子网。如果要创建一组自定义子网，请选择清除子网，然后选择所需的子网。

    > **Important**
    >
    > 计算资源必须通过 VPC 端点或多个公有 IP 地址与 Amazon ECS VPC 端点通信。有关更多信息，请参阅 Amazon ECS 接口 VPC 端点 (AWS PrivateLink)。如果您的实例未配置VPC 端点或公有 IP 地址，则可以使用网络地址转换（NATI）。有关 NAT 的更多信息，请参阅 NAT 网关以及 创建虚拟私有云 。

    c. 对于安全组，选择要与实例关联的 Amazon EC2 安全组。如果要创建一组自定义的安全组，请选择清除安全组。然后，选择您想要的安全组。

22. 选择下一步。

创建作业队列

任务队列会存储您提交的作业，直到 AWS Batch 调度器在您的计算环境中的资源上运行该作业。有关更多信息，请参阅 作业队列。

要为 Amazon EC2 编排创建作业队列，请执行以下操作：

1. 在 作业队列配置部分的名称中，为您的计算环境指定一个唯一的名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
2. 在优先级中，为作业队列输入 0 到 100 之间的整数。

   > **Important**
   >
   > AWS Batch 调度器会为较大的整数值分配更高的优先级。

3. 选择下一步。

创建任务定义

AWS Batch 作业定义指定作业的运行方式。虽然每个作业必须引用作业定义，但可在运行时覆盖作业定义中指定的许多参数。

创建作业定义：

1. 在常规配置部分：

   a. 在名称的常规配置部分，为您的计算环境指定一个唯一的名称。名称长度不超过 128 个字符。名称可以包含大小写字母、数字、连字符（-）和下划线（_）。

   b. （可选）在执行超时中，输入终止未完成作业后的持续时间（以秒为单位）。

   > **Important**
   >
   > 最小超时值为 60 秒。

   c. （可选）标签是为资源分配的标记。要添加标签，请展开标签，然后选择添加标签。输入一个键值对，然后再次选择添加标签。

   > **Important**
   >
   > 如果选择添加标签，则必须输入键值对，然后再次选择添加标签或选择移除标签。

   d. （可选）开启传播标签以将标签传播到 Amazon Elastic Container Service 任务。

2. 在容器配置部分：

   a. 在映像中，输入用于启动容器的映像的名称。默认情况下，Docker Hub 注册表中的所有映像均可用。您也可以使用 repository-url/image:tag 格式指定其他存储库。该参数最长可包含 255 个字符。该参数可以包含大小写字母、数字、连字符（-）、下划线（_）、冒号（:）、句点（.）、正斜杠（/）和数字符号（#）。此参数可映射到 Docker Remote API 的创建容器部分中的 `Image` 和 docker run 的 `IMAGE` 参数。

   > **Note**
   >
   > Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于 Arm 的 Docker 映像只能在基于Arm的计算资源上运行。

- Amazon ECR 公有存储库中的映像使用完整的`registry/repository[:tag]`或`registry/repository[@digest]`命名惯例（例如，`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*）。
- Amazon ECR 存储库中的映像使用完整的`registry/repository:tag`命名惯例（例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*）。
- Docker Hub 上的官方存储库中的映像使用一个名称 (例如，`ubuntu`或`mongo`)。
- Docker Hub 上其他存储库中的映像通过组织名称 (例如，`amazon/amazon-ecs-agent`) 进行限定。
- 其他在线存储库中的映像由域名 (例如，`quay.io/assemblyline/ubuntu`) 进行进一步限定。

b. 对于 Command，指定要传递到容器的命令。此参数映射到 Docker Remote API 创建容器部分中的`Cmd`，以及docker run的`COMMAND`参数。有关 Docker `CMD`参数的更多信息，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。

> **Note**
>
> 您可以在命令中使用参数替代默认值和占位符。有关更多信息，请参阅 参数。

c. （可选）对于执行角色，指定一个 IAM 角色，该角色授予 Amazon ECS 容器代理代表您进行 AWS API 调用的权限。此功能使用 Amazon ECS IAM 角色执行任务。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 任务执行 IAM 角色。

d. （可选）要配置 Job 角色，请选择有权访问 AWS API 的 IAM 角色。此功能使用 Amazon ECS IAM 角色执行任务。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的任务的 IAM 角色。

> **Note**
>
> 此处仅显示具有 Amazon Elastic Container Service Task Role 信任关系的角色。有关为您的 AWS Batch 任务创建 IAM 角色的更多信息，请参阅 A mazon Elastic Container 服务开发者指南中的为任务创建 IAM 角色和策略。

e. （可选）您可以将参数作为键值映射添加到作业定义中，以覆盖作业定义的默认值。若要添加参数：

- 对于参数，选择添加参数。输入键值对，然后再次选择添加参数。

> **Important**
>
> 如果选择添加参数，则必须至少配置一个参数或选择移除参数。

f. 在vCPU的环境配置部分中，指定要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的CpuShares以及docker run的--cpu-shares选项。每个 vCPU 相当于 1024 个 CPU 份额。

g. 对于内存，指定要提供给作业容器的内存硬限制 (以 MiB 为单位)。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的创建容器部分中的Memory以及docker run的--memory选项。

h. 在 GPU 数量中，选择要为容器预留的 GPU 数量。

i. （可选）对于环境变量配置，请选择添加环境变量以添加要传递到容器的环境变量。此参数将映射到 Docker Remote API 的创建容器部分中的Env以及docker run的--env选项。

j. （可选）对于密钥，选择添加密钥，将密钥添加为名称-值对。这些密钥暴露在容器中。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

k. （可选）在 Linux 配置部分中：

   i. 对于 User，输入要在容器内使用的用户名。此参数将映射到 Docker Remote API 的创建容器部分中的User以及docker run的--user选项。

   ii. 要授予作业容器对主机实例（类似于 root 用户）的更高权限，请向右拖动权限滑块。此参数将映射到 Docker Remote API 的创建容器部分中的Privileged以及docker run的--privileged选项。

   iii. 开启启用 Init 处理以在容器内运行 init 进程。该进程转发信号和获得进程。

l. （可选）在文件系统配置部分：

   i. 开启启用只读文件系统以移除对卷的写入权限。

   ii. 在共享内存大小中，输入 /dev/shm 卷的大小（以 MiB 为单位）。

   iii. 在最大交换大小中，输入容器可使用的总交换内存量（以 MiB 为单位）。

   iv. 在 Swappiness 中输入一个介于 0 和 100 之间的值，以指示容器的 swappiness 行为。如果不指定值且启用了交换，则值默认值为 60。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 swappiness。

   v. （可选）展开 其他配置。

vi. 对于 Tmpfs，请选择添加 tmpfs 以添加 `tmpfs` 挂载。

vii. 对于设备，选择添加设备以添加设备：

A. 对于容器路径，指定容器实例中的路径以公开映射到主机实例的设备。如果将其留空，则在容器中使用主机路径。

B. 对于主机路径，指定主机实例中设备的路径。

C. 对于权限，选择要应用于设备的一个或多个权限。可用权限包括读取、写入和 MKNOD。

viii. （可选）对于 Ulimits 配置，请选择添加 ulimit 为容器添加一个 `ulimits` 值。输入名称、软限制和硬限制值，然后选择添加 ulimit。

3. 选择下一步。

创建作业

要创建作业，请执行以下操作：

1. 在作业配置部分的名称中，为该作业指定一个唯一的名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
2. 选择下一步。

审核和创建

在查看和创建页面上，检查配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建资源。

Fargate 入门

AWS Fargate 启动并扩展计算以紧密匹配您为容器指定的资源需求。有了 Fargate，您无需过度配置或为额外的服务器付费。有关更多信息，请参阅 Fargate。

创建计算环境

要为 Fargate 编排创建计算环境，请执行以下操作：

1. 打开 AWS Batch 控制台首次运行向导。

2. 在选择编排类型中，选择 Fargate。
3. 选择下一步。
4. 在名称的计算环境配置部分，为您的计算环境指定唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
5. （可选）标签是为资源分配的标记。要添加标签，请展开标签，然后选择添加标签。输入一个键值对，然后再次选择添加标签。

   > **Important**
   >
   > 如果选择添加标签，则必须输入键值对，然后再次选择添加标签或选择移除标签。

6. （可选）在使用 Fargate Spot 容量的实例配置部分，打开使用竞价型实例启用。
7. 对于vCPU 的最大数量，请输入实例可以使用的最大 vCPU 数量。
8. 在网络配置部分：

   a. 对于虚拟私有云（VPC）ID，选择一个 Amazon VPC。

   b. 对于子网，将列出您的 AWS 账户 子网。如果要创建一组自定义子网，请选择清除子网，然后选择所需的子网。

      > **Important**
      >
      > 计算资源必须通过 VPC 端点或多个公有 IP 地址与 Amazon ECS VPC 端点通信。有关更多信息，请参阅 Amazon ECS 接口 VPC 端点 (AWS PrivateLink)。如果您的实例未配置VPC 端点或公有 IP 地址，则可以使用网络地址转换（NATI）。有关 NAT 的更多信息，请参阅 NAT 网关以及 创建虚拟私有云 。

   c. 对于安全组，选择要与实例关联的 Amazon EC2 安全组。如果要创建一组自定义的安全组，请选择清除安全组。然后，选择您想要的安全组。

9. 选择下一步。

创建作业队列

任务队列会存储您提交的作业，直到 AWS Batch 调度器在您的计算环境中的资源上运行该作业。若要创建作业队列：

要为 Fargate 编排创建作业队列，请执行以下操作：

1. 在 作业队列配置部分的名称中，为您的计算环境指定一个唯一的名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
2. 在优先级中，为作业队列输入 0 到 100 之间的整数。

> **Important**
>
> AWS Batch 调度器会为较大的整数值分配更高的优先级。

3. 选择下一步。

创建任务定义

创建作业定义：

1. 在常规配置部分：
   a. 在名称中，输入自定义作业定义名称。

      在名称的常规配置部分，为您的计算环境指定一个唯一的名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

   b. （可选）在执行超时中，输入终止未完成作业后的持续时间（以秒为单位）。

      > **Important**
      >
      > 最小超时值为 60 秒。

   c. （可选）标签是为资源分配的标记。要添加标签，请展开标签，然后选择添加标签。输入一个键值对，然后再次选择添加标签。

      > **Important**
      >
      > 如果选择添加标签，则必须输入键值对，然后再次选择添加标签或选择移除标签。

   d. （可选）开启传播标签以将标签传播到 Amazon Elastic Container Service 任务。
2. 在 Fargate 平台配置部分中：
   a. （可选）对于 Fargate 平台版本，请输入所需的特定运行时环境。
   b. 对于运行时平台，请选择 LINUX 或 Windows。

c. （仅限 Windows）对于操作系统系列，请选择一个操作系统。

d. 对于 CPU 架构，请选择所需的 CPU 架构。

e. （可选）开启分配公有 IP 以分配公有 IP 地址。

f. 对于临时存储，请输入所需的临时存储量。

> **Note**
>
> 默认情况下，使用 20 GiB 的临时存储空间。要使用额外的临时存储空间，请输入介于 21 GiB 和 100 GiB 之间的值。

g. 对于执行角色，请选择允许亚马逊弹性容器服务 (Amazon ECS) 代理代表 AWS 您拨打电话的任务执行角色。例如，您可以选择 “ecsTaskExecution角色”。

3. 在容器配置部分：

a. 在映像中，输入用于启动容器的映像的名称。默认情况下，Docker Hub 注册表中的所有映像均可用。您也可以使用 repository-url/image:tag 格式指定其他存储库。该参数最长可包含 255 个字符。可以包含大小写字母、数字、连字符（-）、下划线（_）、冒号（:）、句点（.）、正斜杠（/）和数字符号（#）。此参数可映射到 Docker Remote API 的创建容器部分中的 `Image` 和 docker run 的 `IMAGE` 参数。

> **Note**
>
> Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于 Arm 的 Docker 映像只能在基于Arm的计算资源上运行。

- Amazon ECR 公有存储库中的映像使用完整的`registry/repository[:tag]`或`registry/repository[@digest]`命名惯例（例如，`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*）。
- Amazon ECR 存储库中的映像使用完整的`registry/repository:tag`命名惯例（例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*）。
- Docker Hub 上的官方存储库中的映像使用一个名称 (例如，`ubuntu`或`mongo`)。
- Docker Hub 上其他存储库中的映像通过组织名称 (例如，`amazon/amazon-ecs-agent`) 进行限定。

- 其他在线存储库中的映像由域名 (例如，quay.io/assemblyline/ubuntu) 进行进一步限定。

b. 对于 Command，指定要传递到容器的命令。此参数映射到 Docker Remote API 创建容器部分中的Cmd，以及docker run的COMMAND参数。有关 Docker CMD参数的更多信息，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。

> **Note**
> 您可以在命令中使用参数替代默认值和占位符。有关更多信息，请参阅 参数。

> **Tip**
> 选择信息以查看 Bash 和 JSON 代码示例。

c. （可选）您可以将参数作为键值映射添加到作业定义中，以覆盖作业定义的默认值。若要添加参数：

- 对于参数，选择添加参数。输入键值对，然后再次选择添加参数。

> **Important**
> 如果选择添加参数，则必须至少配置一个参数或选择移除参数。

d. （可选）在 Job 角色配置的环境配置部分，选择一个提供使用 AWS API 权限的 IAM 角色。

e. 在vCPU的环境配置部分中，指定要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的CpuShares以及docker run的--cpu-shares选项。每个 vCPU 相当于 1024 个 CPU 份额。

f. 对于内存，指定要提供给作业容器的内存硬限制 (以 MiB 为单位)。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的创建容器部分中的Memory以及docker run的--memory选项。

g. （可选）对于环境变量，请选择添加环境变量以添加要传递到容器的环境变量。此参数将映射到 Docker Remote API 的创建容器部分中的Env以及docker run的--env选项。

4. 选择下一步。

创建作业

要创建 Fargate 作业，请执行以下操作：

1. 在作业配置部分的名称中，为该作业指定一个唯一的名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
2. 选择下一步。

审核和创建

在查看和创建页面上，检查配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建资源。

亚马逊 EK AWS Batch S 入门

AWS Batch On Amazon EKS 是一项托管服务，用于调度批处理工作负载并将其扩展到现有 Amazon EKS 集群中。 AWS Batch 不会代表您创建、管理您的 Amazon EKS 集群或执行其生命周期操作。 AWS Batch 编排向上和向下扩展由这些节点管理的节点 AWS Batch ，并在这些节点上运行 pod。

AWS Batch 不会触及与 Amazon EKS 集群中的 AWS Batch 计算环境无关的节点、自动扩展节点组或 pod 生命周期。 AWS Batch 为了有效运行，其服务相关角色需要在现有 Amazon EKS 集群中具有Kubernetes基于角色的访问控制 (RBAC) 权限。有关更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

AWS Batch 需要一个Kubernetes命名空间，它可以在其中将 pod 限定为 AWS Batch 作业。我们建议使用专用的命名空间将 AWS Batch Pod 与其他集群工作负载隔离开来。

AWS Batch 获得 RBAC 访问权限并建立命名空间后，您可以使用 CreateComputeEnvironmentAPI 操作将该 Amazon EKS 集群关联到 AWS Batch 计算环境。任务队列可以与这个新的 Amazon EKS 计算环境相关联。 AWS Batch 根据 Amazon EKS 任务定义，使用 SubmitJobAPI 操作将任务提交到任务队列。 AWS Batch 然后启动 AWS Batch 托管节点，并将作业队列中的作业作为 Kubernetes pod 放入与 AWS Batch 计算环境关联的 EKS 集群中。

以下各节介绍如何在 Amazon EKS AWS Batch 上进行设置。

目录

先决条件

在开始本教程之前，您必须安装和配置以下工具和资源，以便创建和管理两者 AWS Batch 以及 Amazon EKS 资源。

- AWS CLI – 与 AWS 服务一起使用的命令行工具，包括 Amazon EKS。本指南要求您使用 2.8.6 版或更高版本，或者 1.26.0 版或更高版本。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。安装完成后 AWS CLI，我们建议您也对其进行配置。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 `aws configure` 进行快速配置。
- **`kubectl`** – 用于与 Kubernetes 集群一起使用的命令行工具。本指南要求您使用 `1.23` 版或更高版本。有关更多信息，请参阅《Amazon EKS 用户指南》中的安装或更新 `kubectl`。
- **`eksctl`**— 一款用于处理 Amazon EKS 集群的命令行工具，可自动执行许多单独的任务。本指南要求您使用 `0.115.0` 版或更高版本。有关更多信息，请参阅《Amazon EKS 用户指南》中的安装或更新 `eksctl`。
- 必需的 IAM 权限 — 您使用的 IAM 安全委托人必须具有使用 Amazon EKS IAM 角色和服务关联角色以及 VPC 和相关资源的权限。 AWS CloudFormation有关更多信息，请参阅 IAM 用户指南中的

Amazon Elastic Kubernetes Service 的操作、资源和条件密钥以及使用服务相关角色。您必须以同一用户身份完成本指南中的所有步骤。

- 创建 Amazon EKS 集群 — 有关更多信息，请参阅《亚马逊 EKS 用户指南》eksctl中的 Amazon EKS 入门。

> **Note**
>
> AWS Batch 仅支持具有 API 服务器终端节点的 Amazon EKS 集群，这些终端节点具有公共访问权限，可供公共互联网访问。默认情况下，Amazon EKS 集群 API 服务器终端节点具有公共访问权限。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 集群端点访问控制。

> **Note**
>
> AWS Batch 不为 CoreDNS 或其他部署 pod 提供托管节点编排。如果您需要 CoreDNS，请参阅 Amazon EKS 用户指南中的添加 CoreDNS Amazon EKS 插件。或者，使用 `eksctl create cluster create` 创建集群，默认情况下它包含 CoreDNS。

- 权限 — 调用 CreateComputeEnvironmentAPI 操作来创建使用 Amazon EKS 资源的计算环境的用户需要 `eks:DescribeCluster` API 操作权限。使用使用 Amazon EKS 资源创建计算资源需要同时具有`eks:DescribeCluster`和的权限`eks:ListClusters`。 AWS Management Console

第 1 步：为您的 Amazon EKS 集群做好准备 AWS Batch

必须完成所有步骤。

1. 为 AWS Batch 作业创建专用的命名空间

   用 `kubectl` 以创建新的命名空间。

```
$ namespace=my-aws-batch-namespace
$ cat - <<EOF | kubectl create -f -
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "${namespace}",
```

```
    "labels": {
      "name": "${namespace}"
    }
  }
}
EOF
```

输出：

```
namespace/my-aws-batch-namespace created
```

2. 通过基于角色的访问控制（RBAC）启用访问权限

用于为集群创建Kubernetes角色 AWS Batch 以允许监视节点和 Pod，并用于绑定角色。kubectl您必须为每个 EKS 集群执行一次此操作。

> **Note**
>
> 有关使用 RBAC 授权的更多信息，请参阅用户指南中的使用 RBAC 授权。Kubernetes

```
$ cat - <<EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: aws-batch-cluster-role
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["apps"]
    resources: ["daemonsets", "deployments", "statefulsets", "replicasets"]
    verbs: ["get", "list", "watch"]
```

```
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles", "clusterrolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: aws-batch-cluster-role-binding
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: aws-batch-cluster-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

输出：

```
clusterrole.rbac.authorization.k8s.io/aws-batch-cluster-role created
clusterrolebinding.rbac.authorization.k8s.io/aws-batch-cluster-role-binding created
```

为管理和生命周期 pod 创建命名空间范围 AWS Batch 的Kubernetes角色并将其绑定。必须为每个唯一的命名空间执行一次此操作。

```
$ namespace=my-aws-batch-namespace
$ cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: aws-batch-compute-environment-role
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["create", "get", "list", "watch", "delete", "patch"]
  - apiGroups: [""]
    resources: ["serviceaccounts"]
    verbs: ["get", "list"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
```

```
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: aws-batch-compute-environment-role-binding
  namespace: ${namespace}
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: aws-batch-compute-environment-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

输出：

```
role.rbac.authorization.k8s.io/aws-batch-compute-environment-role created
rolebinding.rbac.authorization.k8s.io/aws-batch-compute-environment-role-binding
 created
```

更新Kubernetesaws-auth配置映射以将前面的 RBAC 权限映射到服务相关角色。 AWS Batch

```
$ eksctl create iamidentitymapping \
    --cluster my-cluster-name \
    --arn "arn:aws:iam::<your-account>:role/AWSServiceRoleForBatch" \
    --username aws-batch
```

输出：

```
2022-10-25 20:19:57 [#]  adding identity "arn:aws:iam::<your-account>:role/
AWSServiceRoleForBatch" to auth ConfigMap
```

> **Note**
>
> 已从服务相关角色的 ARN 中删除路径`aws-service-role/batch.amazonaws.com/`。这是因为 `aws-auth` 配置映射存在问题。有关更多信息，请参阅中带有路径的角色在其 ARN 中包含路径时不起作用。aws-auth configmap

第 2 步：创建 Amazon EKS 计算环境

AWS Batch 计算环境定义计算资源参数以满足您的批处理工作负载需求。在托管计算环境中， AWS Batch 帮助您管理 Amazon EKS 集群中计算资源（Kubernetes节点）的容量和实例类型。这是基于您在创建计算环境时定义的计算资源规范。您可以使用 EC2 按需型实例或 EC2 竞价型实例。

现在，AWSServiceRoleForBatch服务相关角色可以访问您的 Amazon EKS 集群，您可以创建 AWS Batch 资源了。首先，创建一个指向 Amazon EKS 集群的计算环境。

```
$ cat <<EOF > ./batch-eks-compute-environment.json
{
  "computeEnvironmentName": "My-Eks-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:<region>:123456789012:cluster/<cluster-name>",
    "kubernetesNamespace": "my-aws-batch-namespace"
  },
  "computeResources": {
    "type": "EC2",
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "minvCpus": 0,
    "maxvCpus": 128,
    "instanceTypes": [
        "m5"
    ],
    "subnets": [
        "<eks-cluster-subnets-with-access-to-internet-for-image-pull>"
    ],
    "securityGroupIds": [
        "<eks-cluster-sg>"
    ],
    "instanceRole": "<eks-instance-profile>"
  }
```

```
}
EOF
$ aws batch create-compute-environment --cli-input-json file://./batch-eks-compute-environment.json
```

注意事项

- 不应指定该`serviceRole`参数，则将使用 AWS Batch 服务相关角色。 AWS Batch 在 Amazon 上，EKS 仅支持 AWS Batch 服务相关角色。
- Amazon EKS 计算环境仅`BEST_FIT_PROGRESSIVE`支持`SPOT_CAPACITY_OPTIMIZED`、和`SPOT_PRICE_CAPACITY_OPTIMIZED`分配策略。

  > **Note**
  >
  > 建议在大多数情况下使用`SPOT_PRICE_CAPACITY_OPTIMIZED`而不是`SPOT_CAPACITY_OPTIMIZED`。

- 有关 `instanceRole`，请参阅 Amazon EKS 用户指南中的创建 Amazon EKS 节点 IAM 角色和启用 IAM 主体访问您的集群。如果您使用的是容器组（pod）联网，请参阅 Amazon EKS 用户指南中的为 Kubernetes 配置 Amazon VPC CNI 插件以使用服务账户的 IAM 角色。
- 获取 `subnets` 参数的工作子网的一种方法是使用 `eksctl` 创建 Amazon EKS 集群时创建的 Amazon EKS 托管节点组公共子网。否则，请使用具有支持拉取映像的网络路径的子网。
- `securityGroupIds` 参数可以使用与 Amazon EKS 集群相同的安全组。此命令检索集群的安全组 ID。

  ```
  $ eks describe-cluster \
      --name <cluster-name> \
      --query cluster.resourcesVpcConfig.clusterSecurityGroupId
  ```

- Amazon EKS 计算环境的维护是一项共同责任。有关更多信息，请参阅 Kubernetes节点的共同责任。

> **Important**
>
> 在继续操作之前，请务必确认计算环境是否正常。DescribeComputeEnvironmentsAPI 操作可以用来做到这一点。
>
> ```
> $ aws batch describe-compute-environments --compute-environments My-Eks-CE1
> ```

确认 status 参数不是 INVALID。如果是，请查看 statusReason 参数查找原因。有关更多信息，请参阅 故障排除 AWS Batch。

第 3 步：创建作业队列并连接计算环境

```
$ aws batch describe-compute-environments --compute-environments My-Eks-CE1
```

提交到这个新任务队列的任务在加入与您的计算环境关联的 Amazon EKS 集群的 AWS Batch 托管节点上以 Pod 的形式运行。

```
$ cat <<EOF > ./batch-eks-job-queue.json
 {
    "jobQueueName": "My-Eks-JQ1",
    "priority": 10,
    "computeEnvironmentOrder": [
      {
        "order": 1,
        "computeEnvironment": "My-Eks-CE1"
      }
    ]
  }
EOF
$ aws batch create-job-queue --cli-input-json file://./batch-eks-job-queue.json
```

步骤 4：创建作业定义

```
$ cat <<EOF > ./batch-eks-job-definition.json
{
  "jobDefinitionName": "MyJobOnEks_Sleep",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "hostNetwork": true,
      "containers": [
        {
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "sleep",
```

```
            "60"
          ],
          "resources": {
            "limits": {
              "cpu": "1",
              "memory": "1024Mi"
            }
          }
        }
      ],
      "metadata": {
        "labels": {
          "environment": "test"
        }
      }
    }
  }
}
EOF
$ aws batch register-job-definition --cli-input-json file://./batch-eks-job-
definition.json
```

注意事项

- 仅支持单个容器作业。
- cpu 和 memory 参数有一些注意事项。有关更多信息，请参阅 Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项。

第 5 步：提交作业

```
$ aws batch submit-job --job-queue My-Eks-JQ1 \
    --job-definition MyJobOnEks_Sleep --job-name My-Eks-Job1
$ aws batch describe-jobs --job <jobId-from-submit-response>
```

注意事项

- 仅支持单个容器作业。
- 请确保您熟悉 cpu 和 memory 参数的所有相关注意事项。有关更多信息，请参阅 Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项。
- 有关在 Amazon EKS 资源上运行作业的更多信息，请参阅Amazon EKS 作业。

（可选）提交带有优先级的作业

此作业会覆盖传递给容器的命令。

```
$ cat <<EOF > ./submit-job-override.json
{
  "jobName": "EksWithOverrides",
  "jobQueue": "My-Eks-JQ1",
  "jobDefinition": "MyJobOnEks_Sleep",
  "eksPropertiesOverride": {
    "podProperties": {
      "containers": [
        {
          "command": [
            "/bin/sh"
          ],
          "args": [
            "-c",
            "echo hello world"
          ]
        }
      ]
    }
  }
}
EOF
$ aws batch submit-job --cli-input-json file://./submit-job-override.json
```

注意事项

- AWS Batch 在任务完成后积极清理 pod 以将负载减少到。Kubernetes要检查作业的详细信息，必须配置日志记录。有关更多信息，请参阅 使用 CloudWatch Logs 监控 Amazon EKS 作业的 AWS Batch。
- 要提高对操作细节的可见性，请启用 Amazon EKS 控制面板日志记录。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 控制面板日志。
- Daemonsets 和 kubelets 开销会影响可用的 vCPU 和内存资源，特别是扩展和作业布局。有关更多信息，请参阅 Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项。

Amaz AWS Batch on EKS 私有集群入门

AWS Batch 是一项托管服务，可在您的亚马逊 Elastic Kubernetes Service（亚马逊 EKS）集群中协调批处理工作负载。这包括队列、依赖关系跟踪、托管作业重试次数和优先级、Pod 管理和节点扩展。此功能可将您现有的私有 Amazon EKS 集群 AWS Batch 与连接起来，从而大规模运行您的任务。您可以使用 `eksctl`（Amazon EKS 的命令行界面）、 AWS 控制台或创建包含所有其他必要资源的私有 Amazon EKS 集群。AWS Command Line Interface上对私有 Amazon EKS 集群的支持 AWS Batch 通常在商业版中提供（ AWS 区域AWS Batch如果有）。

Amazon EKS 仅限私有集群没有入站/出站互联网访问权限，只有私有子网。Amazon VPC 终端节点用于允许私密访问其他 AWS 服务。 `eksctl`支持使用预先存在的 Amazon VPC 和子网创建完全私有集群。 `eksctl`还会在提供的亚马逊 VPC 中创建 Amazon VPC 终端节点，并修改所提供子网的路由表。

每个子网都应有一个与之关联的显式路由表，因为`eksctl`不会修改主路由表。您的集群必须从您的 Amazon VPC 中的容器注册表中提取映像。此外，您还可以在您的 Amazon VPC 中创建 Amazon 弹性容器注册表，并将容器映像复制到该注册表中，供您的节点从中提取。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。要开始使用 Amazon ECR 私有存储库，请参阅 Amazon ECR 私有存储库。

您可以选择使用 Amazon ECR 创建直通缓存规则。为外部公共注册表创建了拉取缓存规则后，您可以使用您的 Amazon ECR 私有注册表 uriform 资源标识符 (URI) 从该外部公共注册表中提取映像。然后，Amazon ECR 会创建一个存储库并缓存映像。使用 Amazon ECR 私有注册表 URI 提取缓存图像时，Amazon ECR 会检查远程注册表以查看是否有新版本的映像，并最多每 24 小时更新一次您的私有注册表。

目录

先决条件

在开始本教程之前，您必须安装和配置以下工具和资源，以便创建和管理两者 AWS Batch 以及 Amazon EKS 资源。您还需要创建所有必要的资源，包括 VPC、子网、路由表、VPC 终端节点和 Amazon EKS 集群。你需要使用 AWS CLI.

- AWS CLI— 一种用于处理 AWS 服务（包括 Amazon EKS）的命令行工具。本指南要求您使用 2.8.6 版或更高版本，或者 1.26.0 版或更高版本。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。

  安装后 AWS CLI，我们建议您对其进行配置。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 `aws configure` 进行快速配置。
- **`kubectl`**— 用于处理Kubernetes集群的命令行工具。本指南要求您使用 `1.23` 版或更高版本。有关更多信息，请参阅《Amazon EKS 用户指南》中的安装或更新 `kubectl`。
- **`eksctl`**— 一款与 Amazon EKS 集群配合使用的命令行工具，可自动执行许多单独的任务。本指南要求您使用 `0.115.0` 版或更高版本。有关更多信息，请参阅《Amazon EKS 用户指南》中的安装或更新 `eksctl`。
- 必需 AWS Identity and Access Management (IAM) 权限 — 您使用的 IAM 安全主体必须具有使用 Amazon EKS IAM 角色和服务关联角色以及 VPC 和相关资源的权限。 AWS CloudFormation有关更多信息，请参阅 IAM 用户指南中的 Amazon Elastic Kubernetes Service 的操作、资源和条件密钥以及使用服务相关角色。您必须以同一用户身份完成本指南中的所有步骤。
- 创建 Amazon EKS 集群 — 有关更多信息，请参阅《亚马逊 EKS 用户指南》`eksctl`中的 Amazon EKS 入门。

  > **Note**
  >
  > AWS Batch 不为 CoreDNS 或其他部署 pod 提供托管节点编排。如果您需要 CoreDNS，请参阅 Amazon EKS 用户指南中的添加 CoreDNS Amazon EKS 插件。或者，使用 `eksctl create cluster create` 创建集群，默认情况下它包含 CoreDNS。
- 权限 — 调用 CreateComputeEnvironmentAPI 操作来创建使用 Amazon EKS 资源的计算环境的用户需要 `eks:DescribeCluster` API 操作权限。使用使用 Amazon EKS 资源创建计算资源需要同时具有`eks:DescribeCluster`和的权限`eks:ListClusters`。 AWS Management Console
- 使用示例`eksctl`配置文件在 us-east-1 区域创建私有 EK S 集群。

```
kind: ClusterConfig
apiVersion: eksctl.io/v1alpha5
```

```
availabilityZones:
    - us-east-1a
    - us-east-1b
    - us-east-1d
managedNodeGroups:
    privateNetworking: true
privateCluster:
    enabled: true
    skipEndpointCreation: false
```

使用以下命令创建您的资源：`eksctl create cluster -f clusterConfig.yaml`

- 必须将 Batch 托管节点部署到具有所需的 VPC 接口终端节点的子网。有关更多信息，请参阅私有集群要求。

步骤 1：为 EKS 集群做好准备 AWS Batch

必须完成所有步骤。

1. 为 AWS Batch 作业创建专用的命名空间

   用 `kubectl` 以创建新的命名空间。

```
$ namespace=my-aws-batch-namespace
$ cat - <<EOF | kubectl create -f -
{
  "apiVersion": "v1",
  "kind": "Namespace",
  "metadata": {
    "name": "${namespace}",
    "labels": {
      "name": "${namespace}"
    }
  }
}
EOF
```

   输出：

```
namespace/my-aws-batch-namespace created
```

2. 通过基于角色的访问控制（RBAC）启用访问权限

用 `kubectl` 为集群创建 Kubernetes 角色以允许 AWS Batch 监视节点和容器组（pod），并用于绑定该角色。您必须为每个 Amazon EKS 集群执行一次此操作。

> **Note**
>
> 有关使用 RBAC 授权的更多信息，请参阅 Kubernetes 文档中的使用 RBAC 授权。

```
$ cat - <<EOF | kubectl apply -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: aws-batch-cluster-role
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["apps"]
    resources: ["daemonsets", "deployments", "statefulsets", "replicasets"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles", "clusterrolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: aws-batch-cluster-role-binding
subjects:
- kind: User
  name: aws-batch
```

```
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: aws-batch-cluster-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

输出：

```
clusterrole.rbac.authorization.k8s.io/aws-batch-cluster-role created
clusterrolebinding.rbac.authorization.k8s.io/aws-batch-cluster-role-binding created
```

为管理和生命周期 pod 创建命名空间范围 AWS Batch 的Kubernetes角色并将其绑定。必须为每个唯一的命名空间执行一次此操作。

```
$ namespace=my-aws-batch-namespace
$ cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: aws-batch-compute-environment-role
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["create", "get", "list", "watch", "delete", "patch"]
  - apiGroups: [""]
    resources: ["serviceaccounts"]
    verbs: ["get", "list"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: aws-batch-compute-environment-role-binding
  namespace: ${namespace}
subjects:
- kind: User
  name: aws-batch
  apiGroup: rbac.authorization.k8s.io
```

```
roleRef:
  kind: Role
  name: aws-batch-compute-environment-role
  apiGroup: rbac.authorization.k8s.io
EOF
```

输出：

```
role.rbac.authorization.k8s.io/aws-batch-compute-environment-role created
rolebinding.rbac.authorization.k8s.io/aws-batch-compute-environment-role-binding
 created
```

更新Kubernetesaws-auth配置映射以将前面的 RBAC 权限映射到服务相关角色。 AWS Batch

```
$ eksctl create iamidentitymapping \
    --cluster my-cluster-name \
    --arn "arn:aws:iam::<your-account>:role/AWSServiceRoleForBatch" \
    --username aws-batch
```

输出：

```
2022-10-25 20:19:57 [#]  adding identity "arn:aws:iam::<your-account>:role/
AWSServiceRoleForBatch" to auth ConfigMap
```

> **Note**
>
> 已从服务相关角色的 ARN 中删除路径aws-service-role/batch.amazonaws.com/。这是因为 aws-auth 配置映射存在问题。有关更多信息，请参阅中带有路径的角色在其 ARN 中包含路径时不起作用。aws-auth configmap

第 2 步：创建 Amazon EKS 计算环境

AWS Batch 计算环境定义计算资源参数以满足您的批处理工作负载需求。在托管计算环境中， AWS Batch 帮助您管理 Amazon EKS 集群中计算资源（Kubernetes节点）的容量和实例类型。这是基于您在创建计算环境时定义的计算资源规范。您可以使用 EC2 按需型实例或 EC2 竞价型实例。

现在，AWSServiceRoleForBatch服务相关角色可以访问您的 Amazon EKS 集群，您可以创建 AWS Batch 资源了。首先，创建一个指向 Amazon EKS 集群的计算环境。

```
$ cat <<EOF > ./batch-eks-compute-environment.json
{
  "computeEnvironmentName": "My-Eks-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:<region>:123456789012:cluster/<cluster-name>",
    "kubernetesNamespace": "my-aws-batch-namespace"
  },
  "computeResources": {
    "type": "EC2",
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "minvCpus": 0,
    "maxvCpus": 128,
    "instanceTypes": [
        "m5"
    ],
    "subnets": [
        "<eks-cluster-subnets-with-access-to-the-image-for-image-pull>"
    ],
    "securityGroupIds": [
        "<eks-cluster-sg>"
    ],
    "instanceRole": "<eks-instance-profile>"
  }
}
EOF
$ aws batch create-compute-environment --cli-input-json file://./batch-eks-compute-
environment.json
```

注意事项

- 不应指定`serviceRole`参数，则将使用 AWS Batch 服务相关角色。 AWS Batch 在 Amazon 上，EKS 仅支持 AWS Batch 服务相关角色。
- Amazon EKS 计算环境仅`BEST_FIT_PROGRESSIVE`支持`SPOT_CAPACITY_OPTIMIZED`、和`SPOT_PRICE_CAPACITY_OPTIMIZED`分配策略。

> **Note**
>
> 建议在大多数情况下使用`SPOT_PRICE_CAPACITY_OPTIMIZED`而不是`SPOT_CAPACITY_OPTIMIZED`n。

- 有关 `instanceRole`，请参阅 Amazon EKS 用户指南中的创建 Amazon EKS 节点 IAM 角色和启用 IAM 主体访问您的集群。如果您使用的是容器组（pod）联网，请参阅 Amazon EKS 用户指南中的为 Kubernetes 配置 Amazon VPC CNI 插件以使用服务账户的 IAM 角色。
- 获取 `subnets` 参数的工作子网的一种方法是使用 `eksctl` 创建 Amazon EKS 集群时创建的 Amazon EKS 托管节点组公共子网。否则，请使用具有支持拉取映像的网络路径的子网。
- `securityGroupIds` 参数可以使用与 Amazon EKS 集群相同的安全组。此命令检索集群的安全组 ID。

```
$ eks describe-cluster \
    --name <cluster-name> \
    --query cluster.resourcesVpcConfig.clusterSecurityGroupId
```

- Amazon EKS 计算环境的维护是一项共同责任。有关更多信息，请参阅 Amazon EKS 中的安全。

> **Important**
>
> 在继续操作之前，请务必确认计算环境是否正常。DescribeComputeEnvironmentsAPI 操作可用于执行此操作。
>
> ```
> $ aws batch describe-compute-environments --compute-environments My-Eks-CE1
> ```
>
> 确认 `status` 参数不是 `INVALID`。如果是，请查看 `statusReason` 参数查找原因。有关更多信息，请参阅 故障排除 AWS Batch。

第 3 步：创建作业队列并连接计算环境

```
$ aws batch describe-compute-environments --compute-environments My-Eks-CE1
```

提交到这个新任务队列的任务在加入与您的计算环境关联的 Amazon EKS 集群的 AWS Batch 托管节点上以 Pod 的形式运行。

```
$ cat <<EOF > ./batch-eks-job-queue.json
 {
    "jobQueueName": "My-Eks-JQ1",
    "priority": 10,
    "computeEnvironmentOrder": [
      {
```

```
        "order": 1,
        "computeEnvironment": "My-Eks-CE1"
      }
    ]
  }
EOF
$ aws batch create-job-queue --cli-input-json file://./batch-eks-job-queue.json
```

步骤 4：创建作业定义

在作业定义的图像字段中，与其提供指向公共 ECR 存储库中图像的链接，不如提供指向我们私有 ECR 存储库中存储的图像的链接。参见以下作业定义示例：

```
$ cat <<EOF > ./batch-eks-job-definition.json
{
  "jobDefinitionName": "MyJobOnEks_Sleep",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "hostNetwork": true,
      "containers": [
        {
          "image": "account-id.dkr.ecr.region.amazonaws.com/amazonlinux:2",
          "command": [
            "sleep",
            "60"
          ],
          "resources": {
            "limits": {
              "cpu": "1",
              "memory": "1024Mi"
            }
          }
        }
      ],
      "metadata": {
        "labels": {
          "environment": "test"
        }
      }
    }
  }
}
```

```
EOF
$ aws batch register-job-definition --cli-input-json file://./batch-eks-job-
definition.json
```

要运行 kubectl 命令，您需要私有访问您的 Amazon EKS 集群。这意味着所有流向集群 API 服务器的流量都必须来自集群的 VPC 或连接的网络。

第 5 步：提交作业

```
$ aws batch submit-job - -job-queue My-Eks-JQ1 \
    - -job-definition MyJobOnEks_Sleep - -job-name My-Eks-Job1
$ aws batch describe-jobs - -job <jobId-from-submit-response>
```

注意事项

- 仅支持单个容器作业。
- 请确保您熟悉 cpu 和 memory 参数的所有相关注意事项。有关更多信息，请参阅 Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项。
- 有关在 Amazon EKS 资源上运行作业的更多信息，请参阅Amazon EKS 作业。

（可选）提交带有优先级的作业

此作业会覆盖传递给容器的命令。

```
$ cat <<EOF > ./submit-job-override.json
{
  "jobName": "EksWithOverrides",
  "jobQueue": "My-Eks-JQ1",
  "jobDefinition": "MyJobOnEks_Sleep",
  "eksPropertiesOverride": {
    "podProperties": {
      "containers": [
        {
          "command": [
            "/bin/sh"
          ],
          "args": [
            "-c",
            "echo hello world"
          ]
```

```
        }
      ]
    }
  }
}
EOF
$ aws batch submit-job - -cli-input-json file://./submit-job-override.json
```

注意事项

- AWS Batch 在任务完成后积极清理 pod 以将负载减少到。Kubernetes要检查作业的详细信息，必须配置日志记录。有关更多信息，请参阅 使用 CloudWatch Logs 监控 Amazon EKS 作业的 AWS Batch。
- 要提高对操作细节的可见性，请启用 Amazon EKS 控制面板日志记录。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 控制面板日志。
- Daemonsets 和 kubelets 开销会影响可用的 vCPU 和内存资源，特别是扩展和作业布局。有关更多信息，请参阅 Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项。

故障排除

如果由 AWS Batch 启动的节点无权访问存储您的映像的 Amazon ECR 存储库（或任何其他存储库），则您的任务可能会保持 “启动” 状态。这是因为 pod 将无法下载镜像并运行您的 AWS Batch 作业。如果你点击启动的 pod 名称， AWS Batch 你应该能够看到错误消息并确认问题。错误消息应类似于以下内容：

```
Failed to pull image "public.ecr.aws/amazonlinux/amazonlinux:2": rpc error: code =
Unknown desc = failed to pull and unpack image
"public.ecr.aws/amazonlinux/amazonlinux:2": failed to resolve reference
"public.ecr.aws/amazonlinux/amazonlinux:2": failed to do request: Head
"https://public.ecr.aws/v2/amazonlinux/amazonlinux/manifests/2": dial tcp: i/o timeout
```

有关其他常见的故障排除方案，请参阅故障排除 AWS Batch。有关基于 pod 状态的故障排除，请参阅如何对 Amazon EKS 中的容器状态进行故障排除？ 。

作业

工作是从开始的工作单位 AWS Batch。作业可作为在 ECS 集群中的 Amazon ECS 容器实例上运行的容器化应用程序调用。

容器化作业可引用容器映像、命令和参数。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties。

您可以提交大量独立的简单作业。

主题

提交作业

注册作业定义后，可以将其作为作业提交到 AWS Batch 作业队列。在运行时，可以覆盖作业定义中指定的许多参数。

提交作业

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。

2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择作业。
4. 选择 Submit new job。
5. 对于名称，为您的作业定义输入唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
6. 对于 Job definition，为作业选择现有的作业定义。有关更多信息，请参阅 创建单节点作业定义 。
7. 在 作业队列选择现有作业队列。有关更多信息，请参阅 创建作业队列。
8. 对于 作业依赖关系，选择添加作业依赖关系。
   - 在作业 ID 中，输入所有依赖项的作业 ID。然后选择添加作业依赖关系。一个作业最多可有 20 个依赖项。有关更多信息，请参阅 作业依赖项。
9. (仅限数组作业) 对于 Array size，指定一个介于 2 和 10000 之间的数组大小。
10. （可选）展开 标签，然后选择添加标签以向资源添加标签。输入键和可选的值，然后选择添加标签。
11. 选择下一页。
12. 在 作业覆盖部分中：
    a. （可选）在计划优先级中，输入介于 0 和 100 之间的计划优先级值。值越高，优先级越高。
    b. （可选）对于作业尝试，请输入 AWS Batch 尝试将作业移至某一 RUNNABLE 状态的最多次数。您可以输入 1 到 10 之间的数字。有关更多信息，请参阅 自动作业重试。
    c. （可选）对于执行超时，输入超时值（以秒为单位）。执行超时是指未完成的作业终止之前的时间长度。如果某次尝试超过了超时时间，该尝试将停止，状态将转为 FAILED。有关更多信息，请参阅 作业超时。最小值为 60 秒。

       Important

       不要依赖在 Fargate 资源上运行超过 14 天的作业。14 天后，Fargate 资源可能不再可用，该作业很可能会被终止。

    d. （可选）开启 传播标签将标签从作业和作业定义传播到 Amazon ECS 任务。
13. 展开其他配置。
14. （可选）对于重试策略条件，选择退出时添加评估。至少输入一个参数值，然后选择一个操作。对于每组条件，必须将操作设置为重试或退出。这些操作意味着以下几点：

- 重试 — AWS Batch 重试，直到达到您指定的任务尝试次数。
- 退出 — AWS Batch 停止重试作业。

> **Important**
>
> 如果选择退出时添加评估，则至少配置一个参数并选择一个操作或选择退出时移除评估。

15. 对于参数，选择添加参数以添加参数替换占位符。输入一个键和可选的值。
16. 在容器覆盖部分中：

    a. 对于 Command，指定要传递到容器的命令。对于简单的命令，请像输入命令提示符一样输入命令。对于更复杂的命令（例如使用特殊字符），请使用 JSON 语法。

    > **Note**
    >
    > 此参数不能包含空字符串。

    b. 对于 vCPUs，输入要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的`CpuShares`以及docker run的`--cpu-shares`选项。每个 vCPU 相当于 1024 个 CPU 份额。您必须指定至少一个 vCPU。

    c. 对于内存，输入容器可用的内存限制。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的创建容器部分中的`Memory`以及docker run的`--memory`选项。您必须为作业指定至少 4 MiB 内存。

    > **Note**
    >
    > 要最大限度地提高资源利用率，请为特定实例类型的作业确定内存优先级。有关更多信息，请参阅 计算资源内存管理。

    d. （可选）在 GPU 数量中，选择要为容器预留的 GPU 数量。

    e. （可选）对于环境变量，选择添加环境变量以名称-值对的形式添加环境变量。这些变量传递给容器。

    f. 选择下一页。

    g. 对于任务审核，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建作业定义。

作业状态

当您将作业提交到 AWS Batch 作业队列时，该作业将进入SUBMITTED状态。随后，它将经历以下状态，直至成功（退出并返回代码 0）或失败（退出并返回非零代码）。 AWS Batch 任务可具有以下状态：

SUBMITTED

已提交到队列但仍尚未由计划程序评估的任务。计划程序将评估作业，确定在成功完成任何其他作业之前是否有任何未完成的依赖项。如果存在依赖项，作业将进入 PENDING 状态。如果不存在依赖项，作业将进入 RUNNABLE 状态。

PENDING

驻留在队列中但因依赖其他作业或资源而导致尚无法运行的作业。在满足依赖关系后，作业将进入 RUNNABLE 状态。

RUNNABLE

驻留在队列中的没有任何未完成依赖项的作业，可在主机中计划运行该作业。一旦映射到作业队列的某个计算环境提供足够的资源，处于此状态的作业就会启动。不过，当没有足够资源可用时，作业会无限期地保持此状态。

> **Note**
>
> 如果您的任务未进行到 STARTING，请参阅故障排除部分中的 作业在RUNNABLE状态卡住。

STARTING

已在主机上计划运行这些作业，并且相关的容器启动操作正在进行中。在提取容器映像并且容器已启动并运行后，作业将过渡到 RUNNING 状态。

图像提取持续时间、Amazon EKS initContainer 完成时长和 Amazon ECS 容器依赖解析持续时间处于起始状态。为作业提取图像所花费的时间等于您的作业处于 “开始” 状态的时间量。

例如，如果提取作业的图像需要三分钟，则您的作业将在三分钟内处于 “启动” 状态。如果 initContainers 总共需要十分钟才能完成，那么你的 Amazon EKS 任务将在 “启动” 状态下持续十分钟。如果您的 Amazon ECS 任务中设置了 Amazon ECS ContainerDependencies，则在解决所有容器依赖关系（其运行时间）之前，该任务将处于启动状态。启动不包含在超时中；持续时间从 RUNNING 开始。有关更多信息，请参阅 Job 状态。

RUNNING

作业正作为容器作业在计算环境中的 Amazon ECS 容器实例上运行。当作业容器退出时，进程退出代码将确定作业是成功还是失败。退出代码 0 表示成功，非零退出代码表示失败。如果作业与失败的尝试关联，但在其可选重试策略配置中还有剩余的尝试次数，则作业将再次进入 RUNNABLE 状态。有关更多信息，请参阅 自动作业重试。

> **Note**
>
> RUNNING作业日志可在 CloudWatch 日志中找到。日志组是 /aws/batch/job，日志流名称格式如下：*first200CharsOfJobDefinitionName*/default/*ecs_task_id*。这种格式未来可能会改变。
> 任务达到RUNNING状态后，您可以通过 DescribeJobsAPI 操作以编程方式检索其日志流名称。有关更多信息，请参阅 Amazon Logs 用户指南中的查看发送到 CloudWatch CloudWatch 日志的日志数据。默认情况下，这些日志永不过期。但是，您可以修改备份保留期。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南中的更改 CloudWatch 日志数据保留期。

SUCCEEDED

作业已成功完成，并返回退出代码 0。作业的SUCCEEDED作业状态会持续 AWS Batch 至少 7 天。

> **Note**
>
> SUCCEEDED作业日志可在 CloudWatch 日志中找到。日志组是 /aws/batch/job，日志流名称格式如下：*first200CharsOfJobDefinitionName*/default/*ecs_task_id*。这种格式未来可能会改变。
> 任务达到RUNNING状态后，您可以通过 DescribeJobsAPI 操作以编程方式检索其日志流名称。有关更多信息，请参阅 Amazon Logs 用户指南中的查看发送到 CloudWatch CloudWatch 日志的日志数据。默认情况下，这些日志永不过期。但是，您可以修改备份保留期。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南中的更改 CloudWatch 日志数据保留期。

FAILED

在执行所有可用尝试后，作业失败。FAILED 作业的作业状态在 AWS Batch 中保留至少 7 天。

> **Note**
>
> FAILED作业日志可在 CloudWatch 日志中找到。日志组是 /aws/batch/job，日志流名称格式如下：*first200CharsOfJobDefinitionName*/default/*ecs_task_id*。这种格式未来可能会改变。
> 任务达到RUNNING状态后，您可以通过 DescribeJobsAPI 操作以编程方式检索其日志流。有关更多信息，请参阅 Amazon Logs 用户指南中的查看发送到 CloudWatch CloudWatch 日志的日志数据。默认情况下，这些日志永不过期。但是，您可以修改备份保留期。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南中的更改 CloudWatch 日志数据保留期。

AWS Batch 作业环境变量

AWS Batch 在容器作业中设置特定的环境变量。这些环境变量为作业中的容器提供了内省能力。您可以在应用程序的逻辑中使用这些变量的值。所有 AWS Batch 设置的变量都以AWS_BATCH_前缀开头。这是受保护的环境变量前缀。在作业定义或覆盖中，您不能将此前缀用于自己的变量。

以下环境变量在作业容器中可用：

`AWS_BATCH_CE_NAME`

此变量设置为您的作业所在的计算环境的名称。

`AWS_BATCH_JOB_ARRAY_INDEX`

此变量仅在子数组作业中设置。数组作业索引从 0 开始，并且每个子作业接收一个唯一索引编号。例如，包含 10 个子级的数组作业具有索引值 0-9。您可以使用此索引值来控制数组作业子级的差异。有关更多信息，请参阅 教程：使用数组作业索引控制作业差异化。

`AWS_BATCH_JOB_ARRAY_SIZE`

此变量设置为父数组作业的大小。父数组作业的大小在此变量中传递给子数组作业。

`AWS_BATCH_JOB_ATTEMPT`

此变量设置为作业尝试次数。第一次尝试编号为 1。有关更多信息，请参阅 自动作业重试。

`AWS_BATCH_JOB_ID`

此变量设置为作 AWS Batch 业 ID。

AWS_BATCH_JOB_KUBERNETES_NODE_UID

此变量被设置为运行该容器的 Kubernetes 集群中的节点对象的 Kubernetes UID。此变量仅适用于在 Amazon EKS 资源上运行的作业。有关更多信息，请参阅 Kubernetes文档 中的 UIDs。

AWS_BATCH_JOB_MAIN_NODE_INDEX

此变量仅在多节点并行作业中设置。此变量设置为作业的主节点的索引号。您的应用程序代码可以将 AWS_BATCH_JOB_MAIN_NODE_INDEX 与单个节点上的 AWS_BATCH_JOB_NODE_INDEX 进行比较，以确定它是否为主节点。

AWS_BATCH_JOB_MAIN_NODE_PRIVATE_IPV4_ADDRESS

此变量仅在多节点并行作业子节点中设置。此变量设置不出现在主节点中，但设置为作业的主节点的私有 IPv4 地址。您的子节点的应用程序代码可以使用此地址与主节点进行通信。

AWS_BATCH_JOB_NODE_INDEX

此变量仅在多节点并行作业中设置。此变量设置为节点的节点索引号。节点索引从 0 开始，并且每个节点接收一个唯一的索引号。例如，包含 10 个子级的多节点并行作业具有索引值 0-9。

AWS_BATCH_JOB_NUM_NODES

此变量仅在多节点并行作业中设置。此变量设置为您为多节点并行作业请求的节点数。

AWS_BATCH_JQ_NAME

此变量将设置为您的作业提交到的作业队列的名称。

自动作业重试

您可以将重试策略应用于作业和作业定义，这将允许失败的作业自动重试。可能的失败场景包括以下情况：

- 来自容器作业的任何非零退出代码
- Amazon EC2 实例失败或终止
- 内部 AWS 服务错误或中断

在将作业提交到作业队列并置于 RUNNING 状态时，将视为一次尝试。默认情况下，每个作业均可尝试移至 SUCCEEDED 或 FAILED 作业状态一次。不过，作业定义和作业提交工作流都可以用来指定一个具有 1 至 10 次尝试的重试策略。如果指定了 OnExit evaluate，则它最多可以包含 5 个重试策略。如果指定了 OnExit evaluate，但所有重试策略都不匹配，则会重试该作业。对于与退出不匹配的作业，

请添加因任何原因退出的最终条目。例如，此 evaluateOnExit 对象有两个操作为 RETRY 的条目，和一个操作为 EXIT 的最后一个条目。

```
"evaluateOnExit": [
    {
        "action": "RETRY",
        "onReason": "AGENT"
    },
    {
        "action": "RETRY",
        "onStatusReason": "Task failed to start"
    },
    {
        "action": "EXIT",
        "onReason": "*"
    }
]
```

在运行时，AWS_BATCH_JOB_ATTEMPT 环境变量将设置为容器的相应作业尝试次数。第一次尝试的编号为 1，后续尝试的编号按升序排列 (2、3、4，以此类推)。

例如，假设作业尝试因任何原因失败，并且重试配置中指定的尝试次数大于 AWS_BATCH_JOB_ATTEMPT 数。则该作业被放回 RUNNABLE 状态。有关更多信息，请参阅 作业状态。

> **Note**
>
> 不会重试已取消或终止的作业。此外，也不会重试因作业定义无效而导致失败的作业。

有关更多信息，请参阅 重试策略、创建单节点作业定义 、提交作业 和已停止的任务错误代码。

作业依赖项

提交 AWS Batch 作业时，您可以指定该作业所依赖的作业 ID。在执行此操作时， AWS Batch 计划程序将确保您的作业仅在指定的依赖项已成功完成后运行。成功后，依赖性作业将从 PENDING 转换到 RUNNABLE，然后再转换到 STARTING 和 RUNNING。如果任何作业依赖项失败，则依赖性作业会自动从 PENDING 转换到 FAILED。

例如，作业 A 可依赖于最多 20 个作业，这些作业必须成功，然后才能运行作业 A。然后，您可以提交依赖于作业 A 的其他作业，最多 19 个其他作业。

对于数组作业，您可以指定 SEQUENTIAL 类型依赖项，而无需指定作业 ID，以便每个子数组作业按顺序完成 (从索引 0 开始)。您也可以使用作业 ID 指定 N_TO_N 类型依赖项。这样一来，此作业的每个子索引必须等待每个依赖项的相应子索引完成后才能开始。有关更多信息，请参阅 数组作业。

要提交具有依赖关系的 AWS Batch 作业，请参阅提交作业。

作业超时

可以为作业配置超时时间，以便在某个作业运行的时间超过超时时间时让 AWS Batch 终止该作业。例如，您可能有一个作业，并且您知道该作业只需 15 分钟即可完成。有时，您的应用程序会陷入循环并永远运行，因此您可以将超时设置为 30 分钟以终止卡住的作业。

> **Important**
>
> 默认情况下， AWS Batch 没有任务超时。如果您未定义作业超时，则作业将一直运行到容器退出。

您可以指定一个 `attemptDurationSeconds` 参数，该参数必须至少为 60 秒，在您的任务定义中，或者在您提交任务时。在任务尝试的时间`startedAt`戳之后经过此秒数后， AWS Batch 将终止该作业。在计算资源上，作业的容器会收到 SIGTERM 信号，以便为应用程序提供正常关闭的机会。如果容器在 30 秒后仍在运行，则会发送 SIGKILL 信号以强制关闭容器。

超时终止是基于最佳效果来处理的。您不应期望超时终止正好在作业尝试超时时执行（可能有几秒钟的延迟）。如果您的应用程序需要精确的超时执行，您应该在该应用程序中实施此逻辑。如果您有大量任务同时超时，超时终止的行为将类似于先入先出队列，在此队列中，任务是成批终止的。

> **Note**
>
> AWS Batch 作业没有最大超时值。

如果某个任务因超过超时时间而终止，它不会被重试。如果任务尝试自行失败，则当启用了重试并且超时倒计时对新尝试重新开始时，该任务可能会重试。

> **Important**
>
> 在 Fargate 资源上运行的作业不能期望运行超过 14 天。如果超时时间超过 14 天，Fargate 资源可能不再可用，作业将被终止。

对于数组任务，子任务与父任务具有相同的超时配置。

有关提交带有超时配置的 AWS Batch 作业的信息，请参阅提交作业。

Amazon EKS 作业

作业是最小的工作单位 AWS Batch。Amazon EKS 上的 AWS Batch 作业具有到Kubernetes容器的 one-to-one 映射。 AWS Batch 作业定义是 AWS Batch 作业的模板。提交 AWS Batch 作业时，您可以引用作业定义、定位作业队列并提供作业名称。在 Amazon EKS 上 AWS Batch 作业的任务定义中，eksProper ties 参数定义了 AWS Batch 亚马逊 EKS 上作业支持的一组参数。在SubmitJob请求中，eks PropertiesOverride 参数允许覆盖某些常用参数。这样，您就可以为多个作业使用作业定义模板。将任务分派到您的 Amazon EKS 集群时，会将该任务 AWS Batch 转换为 podspec (Kind: Pod)。podspec使用一些附加 AWS Batch 参数来确保作业的扩展和调度正确。 AWS Batch 结合标签和污点，确保作业仅在 AWS Batch 托管节点上运行，而其他 pod 不会在这些节点上运行。

> **Important**
>
> - 如果未在 Amazon EKS 任务定义中明确设置该hostNetwork参数，则 AWS Batch 默认情况下的 pod 联网模式为主机模式。更具体地说，将应用以下设置：hostNetwork=true 和 dnsPolicy=ClusterFirstWithHostNet。
> - AWS Batch 在 pod 完成任务后立即清理任务窗格。要查看容器组（pod）应用程序日志，请为您的集群配置日志服务。有关更多信息，请参阅 使用 CloudWatch Logs 监控 Amazon EKS 作业的 AWS Batch。

将正在运行的作业映射到容器组（pod）和节点

正在运行的作业的 podProperties 具有为当前作业尝试设置的 podName 参数和 nodeName 参数。使用 DescribeJobsAPI 操作查看这些参数。

下面是示例输出。

```
$ aws batch describe-jobs --job 2d044787-c663-4ce6-a6fe-f2baf7e51b04
{
 "jobs": [
  {
   "status": "RUNNING",
   "jobArn": "arn:aws:batch:us-east-1:123456789012:job/2d044787-c663-4ce6-a6fe-
f2baf7e51b04",
```

```
   "jobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/
MyJobOnEks_SleepWithRequestsOnly:1",
   "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/My-Eks-JQ1",
   "jobId": "2d044787-c663-4ce6-a6fe-f2baf7e51b04",
   "eksProperties": {
    "podProperties": {
     "nodeName": "ip-192-168-55-175.ec2.internal",
     "containers": [
      {
       "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
       "resources": {
        "requests": {
         "cpu": "1",
         "memory": "1024Mi"
        }
       }
      }
     ],
     "podName": "aws-batch.b0aca953-ba8f-3791-83e2-ed13af39428c"
    }
   }
  }
 ]
}
```

对于启用了重试功能的作业，DescribeJobsAPI 操作nodeName的eksAttempts列表参数中包含每次已完成尝试的podName和。当前运行尝试的 podName 和 nodeName 在 podProperties 对象中。

如何将正在运行的容器组（pod）映射回其作业

Pod uuid 的标签表示它所属的计算环境的jobId和。 AWS Batch 注入环境变量，以便作业的运行时可以引用作业信息。有关更多信息，请参阅 AWS Batch 作业环境变量。您可以运行以下命令以查看此信息。输出如下所示。

```
$ kubectl describe pod aws-batch.14638eb9-d218-372d-ba5c-1c9ab9c7f2a1 -n my-aws-batch-
namespace
Name:         aws-batch.14638eb9-d218-372d-ba5c-1c9ab9c7f2a1
Namespace:    my-aws-batch-namespace
Priority:     0
Node:         ip-192-168-45-88.ec2.internal/192.168.45.88
Start Time:   Wed, 26 Oct 2022 00:30:48 +0000
Labels:       batch.amazonaws.com/compute-environment-uuid=5c19160b-
d450-31c9-8454-86cf5b30548f
```

```
              batch.amazonaws.com/job-id=f980f2cf-6309-4c77-a2b2-d83fbba0e9f0
              batch.amazonaws.com/node-uid=a4be5c1d-9881-4524-b967-587789094647
...
Status:       Running
IP:           192.168.45.88
IPs:
  IP:  192.168.45.88
Containers:
  default:
    Image:         public.ecr.aws/amazonlinux/amazonlinux:2
    ...
    Environment:
      AWS_BATCH_JOB_KUBERNETES_NODE_UID:  a4be5c1d-9881-4524-b967-587789094647
      AWS_BATCH_JOB_ID:                   f980f2cf-6309-4c77-a2b2-d83fbba0e9f0
      AWS_BATCH_JQ_NAME:                  My-Eks-JQ1
      AWS_BATCH_JOB_ATTEMPT:              1
      AWS_BATCH_CE_NAME:                  My-Eks-CE1

...
```

AWS Batch Amazon EKS 工作支持的功能

以下是在 Amazon EKS 上运行的Kubernetes作业也很常见的 AWS Batch 特定功能：

- 作业依赖项
- 数组作业
- 作业超时
- 自动作业重试
- 公平份额调度

Kubernetes**Secrets** 和 **ServiceAccounts**

AWS Batch 支持引用KubernetesSecrets和。ServiceAccounts您可以配置容器组（pod）将 Amazon EKS IAmazon EKS IAM 角色用于服务账户。有关更多信息，请参阅 Amazon EKS 用户指南中的将容器组（pod）配置为使用Kubernetes服务账户。

相关文档

- Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项
- 在 Amazon EKS 资源上创建基于 GPU 的任务

- 作业在RUNNABLE状态卡住

数组作业

数组作业是共享通用参数 (如作业定义、vCPU 和内存) 的作业。它会以一系列相关但独立的基本作业的形式运行，这些作业可能跨多个主机分布，而且可能同时运行。数组作业是运行高度并行作业 (如 Monte Carlo 模拟、参数扫描或大型渲染作业) 的最高效方式。

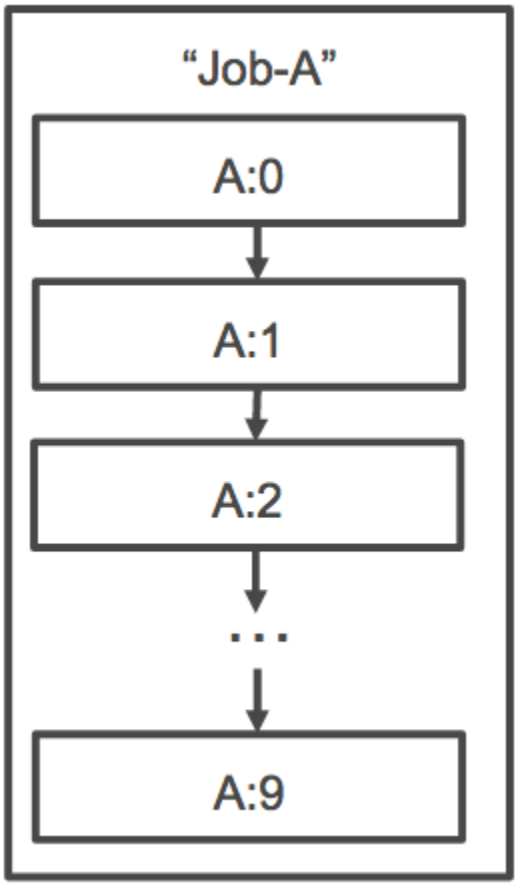
AWS Batch 阵列作业的提交方式与普通作业一样。但是，您可以指定一个数组大小 (介于 2 和 10000 之间) 来定义该数组中应运行的子作业数。如果您提交一个数组大小为 1000 的作业，则单个作业会运行并生成 1000 个子作业。该数组作业是用于管理所有子作业的参考或指针。这种方式将允许您使用单个查询提交大型工作负载。`attemptDurationSeconds` 参数中指定的超时适用于每个子作业。父阵列作业没有超时。

当您提交阵列作业时，父阵列作业将获得一个普通的 AWS Batch 作业 ID。每个子作业都有相同的基本 ID。但是，子作业的数组索引将附加到父 ID 的末尾，如数组的首个子作业的 *`example_job_ID`*`:0`。

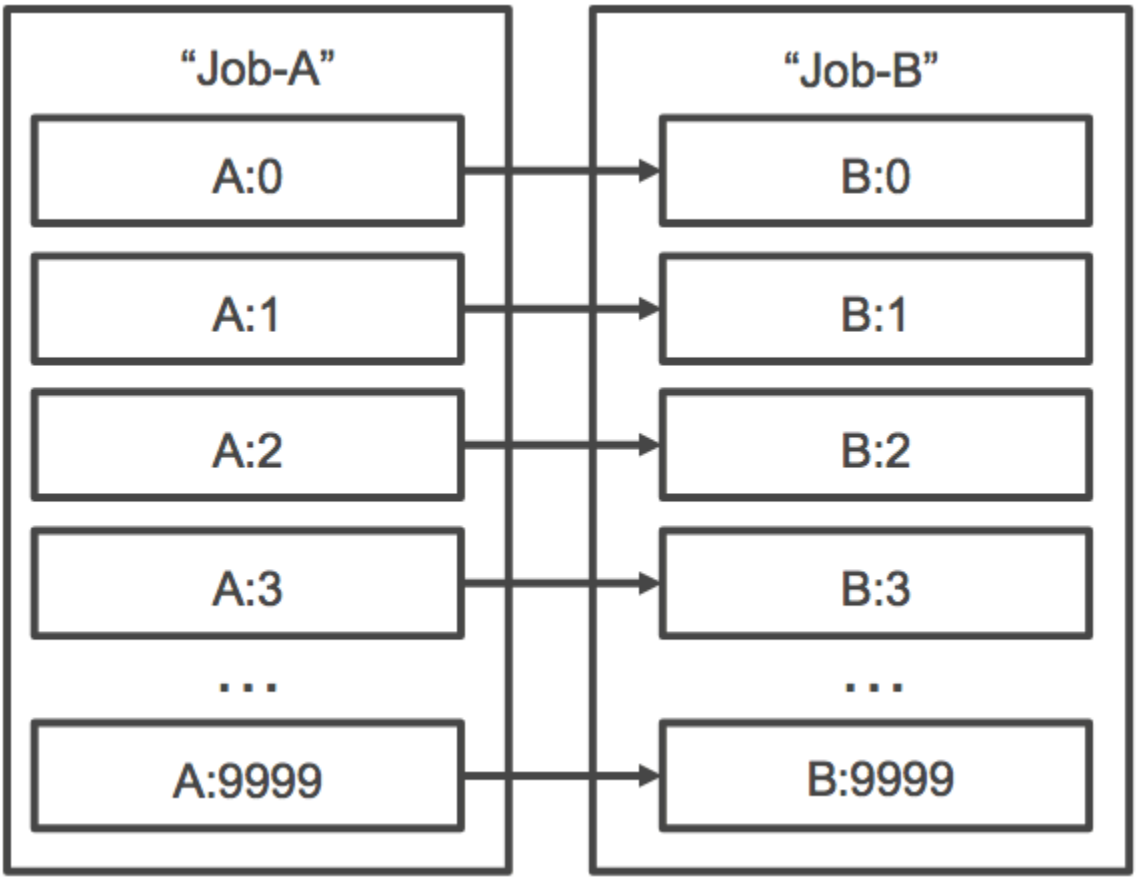
父阵列作业可以输入 SUBMITTED、PENDING、FAILED 或 SUCCEEDED 状态。当任何子作业更新为 RUNNABLE时，阵列父作业将更新为 PENDING。有关作业依赖关系的更多信息，请参阅 作业依赖项。

在运行时，AWS_BATCH_JOB_ARRAY_INDEX 环境变量将设置为容器的相应作业数组索引编号。第一个数组作业索引的编号为 0，后续尝试的编号按升序排列 (1、2、3，依此类推)。您可以使用此索引值来控制数组作业子级的差异。有关更多信息，请参阅 教程：使用数组作业索引控制作业差异化。

对于数组作业依赖项，您可以为依赖项指定一种类型，如 SEQUENTIAL 或 N_TO_N。您可以指定 SEQUENTIAL 类型依赖项 (无需指定作业 ID)，以便每个子数组作业按顺序完成 (从索引 0 开始)。例如，如果您提交一个数组大小为 100 的数组作业，并指定类型为 SEQUENTIAL 的依赖项，则会按顺序生成 100 个子作业，其中，首个子作业必须先成功，然后才能开始下一个子作业。下图显示作业 A，它是一个数组大小为 10 的数组作业。作业 A 的子索引中的每个作业均依赖于其前一个子作业。作业 A:1 无法启动，直到作业 A:0 完成。



您也可以指定 N_TO_N 类型依赖项，并指定数组任务的任务 ID 。这样一来，此作业的每个子索引必须等待每个依赖项的相应子索引完成后才能开始。下图显示作业 A 和作业 B，二者是数组大小均为 10,000 的数组作业。作业 B 的子索引中的每个作业均依赖于作业 A 中的相应索引。作业 B:1 无法开始，直到作业 A:1 完成。



如果您取消或终止父数组作业，则其所有子作业将随它一起取消或终止。您可以取消或终止单个子作业 (这会将它们迁移至 FAILED 状态)，而不会影响其他子作业。但是，如果子数组作业失败 (自行或通过手动取消或终止)，则父作业也会失败。

示例数组作业工作流

AWS Batch 客户的常见工作流程是运行先决条件设置任务，对大量输入任务运行一系列命令，然后完成一项任务，该任务汇总结果并将摘要数据写入 Amazon S3、DynamoDB、Amazon Redshift 或 Aurora。

例如：

- JobA：一个标准的非数组任务，它对 Amazon S3 存储桶 BucketA 中的对象执行快速列示和元数据验证。SubmitJobJSON 语法如下所示。

```
{
    "jobName": "JobA",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobA-list-and-validate:1"
```

```
}
```

- JobB：一个包含 10000 个副本的数组作业，它依赖于 JobA，针对 BucketA 中的每个对象运行 CPU 密集型命令并将结果上传至 BucketB。SubmitJobJSON 语法如下所示。

```
{
    "jobName": "JobB",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobB-CPU-Intensive-Processing:1",
    "containerOverrides": {
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "4096"
            },
            {
                "type": "VCPU",
                "value": "32"
            }
        ]
   }
    "arrayProperties": {
        "size": 10000
    },
    "dependsOn": [
        {
            "jobId": "JobA_job_ID"
  }
    ]
}
```

- JobC：另一个包含 10000 个副本的数组任务，依赖于 JobB，具有 N_TO_N 依赖项模型，针对 BucketB 中的每个项目运行内存密集型命令，将元数据写入到 DynamoDB，并将生成的输出上传至 BucketC。SubmitJobJSON 语法如下所示。

```
{
    "jobName": "JobC",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobC-Memory-Intensive-Processing:1",
    "containerOverrides": {
        "resourceRequirements": [
            {
                "type": "MEMORY",
```

```
                "value": "32768"
            },
            {
                "type": "VCPU",
                "value": "1"
            }
        ]
    }
     "arrayProperties": {
         "size": 10000
     },
     "dependsOn": [
         {
             "jobId": "JobB_job_ID",
             "type": "N_TO_N"
         }
     ]
}
```

- JobD：一个执行 10 个验证步骤的数组任务，其中每个步骤需要查询 DynamoDB 并且可能与上述任一 Amazon S3 存储桶交互。JobD 中的每个步骤都运行相同的命令。但是，该行为因作业容器内的 AWS_BATCH_JOB_ARRAY_INDEX 环境变量的值而异。这些验证步骤按顺序运行 (例如，先运行 JobD:0，再运行 JobD:1，依此类推)。SubmitJobJSON 语法如下所示。

```
{
    "jobName": "JobD",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobD-Sequential-Validation:1",
    "containerOverrides": {
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "32768"
            },
            {
                "type": "VCPU",
                "value": "1"
            }
        ]
    }
     "arrayProperties": {
         "size": 10
     },
```

```
    "dependsOn": [
        {
            "jobId": "JobC_job_ID"
        },
        {
            "type": "SEQUENTIAL"
        },

    ]
}
```

- JobE：一个最终的非数组任务，它执行一些简单的清除操作并发送包含管道已完成的消息和指向输出 URL 的链接的 Amazon SNS 通知。SubmitJobJSON 语法如下所示。

```
{
    "jobName": "JobE",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobE-Cleanup-and-Notification:1",
    "parameters": {
        "SourceBucket": "s3://JobD-Output-Bucket",
        "Recipient": "pipeline-notifications@mycompany.com"
    },
    "dependsOn": [
        {
            "jobId": "JobD_job_ID"
        }
    ]
}
```

教程：使用数组作业索引控制作业差异化

本教程介绍如何使用 AWS_BATCH_JOB_ARRAY_INDEX 环境变量来区分子作业。每个子作业都分配给这个变量。该示例使用子作业的索引号来读取文件中的特定行。然后，它将与该行号关联的参数替换为作业容器内的命令。结果是，你可以有多个运行相同的 Docker 镜像和命令参数的 AWS Batch 作业。但是，结果有所不同，因为使用数组作业索引作为修饰符。

在本教程中，您将创建一个文本文件，该文件包含了彩虹的所有颜色，并且每种颜色单独占一行。然后，您将为 Docker 容器创建一个入口点脚本，该脚本会将索引转换为可用于颜色文件中的行号的值。索引从零开始，但行号从一开始。创建一个 Dockerfile，该文件会将颜色和索引文件复制到容器映像并将该映像的 ENTRYPOINT 设置为入口点脚本。Dockerfile 和资源将生成为一个 Docker 映像并被推送

到 Amazon ECR。然后，您注册一个使用您的新容器镜像的作业定义，提交包含该作业定义的 AWS Batch 数组作业，然后查看结果。

先决条件

本教程包含以下先决条件：

- AWS Batch 计算环境。有关更多信息，请参阅 创建计算环境。
- AWS Batch 作业队列和相关的计算环境。有关更多信息，请参阅 创建作业队列。
- 已 AWS CLI 安装在您的本地系统上。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。
- 在本地系统上安装的 Docker。有关更多信息，请参阅 Docker 文档中的关于 Docker CE。

步骤 1：构建容器映像

可以在命令参数中使用作业定义中的 AWS_BATCH_JOB_ARRAY_INDEX。但是，我们建议您创建容器映像，该映像改用 Entrypoint 脚本中的变量。本节介绍如何创建此类容器映像。

构建 Docker 容器映像

1. 创建要用作 Docker 映像工作区的新目录并导航到该目录。
2. 在工作区目录中创建一个名为 colors.txt 的文件，并将下面的内容粘贴到其中。

```
red
orange
yellow
green
blue
indigo
violet
```

3. 在工作区目录中创建一个名为 print-color.sh 的文件，并将下面的内容粘贴到其中。

> **Note**
>
> LINE 变量被设置为 AWS_BATCH_JOB_ARRAY_INDEX + 1，因为数组索引从 0 开始，但行号从 1 开始。COLOR 变量被设置为与其行号关联的 colors.txt 中的颜色。

```
#!/bin/sh
LINE=$((AWS_BATCH_JOB_ARRAY_INDEX + 1))
COLOR=$(sed -n ${LINE}p /tmp/colors.txt)
echo My favorite color of the rainbow is $COLOR.
```

4. 在工作区目录中创建一个名为 Dockerfile 的文件，并将下面的内容粘贴到其中。此 Dockerfile 会将以前的文件复制到您的容器并设置要在容器启动时运行的入口点脚本。

```
FROM busybox
COPY print-color.sh /tmp/print-color.sh
COPY colors.txt /tmp/colors.txt
RUN chmod +x /tmp/print-color.sh
ENTRYPOINT /tmp/print-color.sh
```

5. 生成 Docker 映像。

```
$ docker build -t print-color .
```

6. 使用以下脚本测试容器。此脚本在本地将 AWS_BATCH_JOB_ARRAY_INDEX 变量设置为 0，然后递增它以模拟具有 7 个子级的数组作业的行为。

```
$ AWS_BATCH_JOB_ARRAY_INDEX=0
while [ $AWS_BATCH_JOB_ARRAY_INDEX -le 6 ]
do
    docker run -e AWS_BATCH_JOB_ARRAY_INDEX=$AWS_BATCH_JOB_ARRAY_INDEX print-color
    AWS_BATCH_JOB_ARRAY_INDEX=$((AWS_BATCH_JOB_ARRAY_INDEX + 1))
done
```

下面是输出。

```
My favorite color of the rainbow is red.
My favorite color of the rainbow is orange.
My favorite color of the rainbow is yellow.
My favorite color of the rainbow is green.
My favorite color of the rainbow is blue.
My favorite color of the rainbow is indigo.
My favorite color of the rainbow is violet.
```

步骤 2：推送映像到 Amazon ECR

既然您已构建并测试 Docker 容器，您必须将其推送到映像存储库。此示例使用 Amazon ECR，但您可以使用其他注册表，例如 DockerHub。

1. 创建用于存储您的容器映像的 Amazon ECR 映像存储库。此示例仅使用 AWS CLI，但您也可以使用 AWS Management Console。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南 中的 创建存储库。

   ```
   $ aws ecr create-repository --repository-name print-color
   ```

2. 用从上一步返回的 Amazon ECR r存储库 URI 标记 `print-color` 映像。

   ```
   $ docker tag print-color aws_account_id.dkr.ecr.region.amazonaws.com/print-color
   ```

3. 登录到您的 Amazon ECR 注册表。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 | docker login \
       --username AWS \
       --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
   ```

4. 将映像推送到 Amazon ECR。

   ```
   $ docker push aws_account_id.dkr.ecr.region.amazonaws.com/print-color
   ```

步骤 3：创建并注册作业定义

现在，您的 Docker 镜像已在镜像注册表中，您可以在 AWS Batch 作业定义中对其进行指定。则您可以稍后使用它来运行数组作业。此示例只使用 AWS CLI。但是，您还可以使用 AWS Management Console。有关更多信息，请参阅 创建单节点作业定义 。

创建作业定义

1. 在工作区目录中创建一个名为 `print-color-job-def.json` 的文件，并将下面的内容粘贴到其中。将映像存储库 URI 替换为您自己的映像的 URI。

   ```
   {
     "jobDefinitionName": "print-color",
   ```

```
  "type": "container",
  "containerProperties": {
    "image": "aws_account_id.dkr.ecr.region.amazonaws.com/print-color",
    "resourceRequirements": [
        {
            "type": "MEMORY",
            "value": "250"
        },
        {
            "type": "VCPU",
            "value": "1"
        }
    ]
  }
}
```

2. 向注册作业定义 AWS Batch。

```
$ aws batch register-job-definition --cli-input-json file://print-color-job-
def.json
```

步骤 4：提交 AWS Batch 阵列作业

注册任务定义后，您可以提交使用新容器映像的 AWS Batch 数组作业。

提交 AWS Batch 阵列作业

1. 在工作区目录中创建一个名为 `print-color-job.json` 的文件，并将下面的内容粘贴到其中。

> **Note**
>
> 此示例使用本 the section called “先决条件” 节中提到的作业队列。

```
{
  "jobName": "print-color",
  "jobQueue": "existing-job-queue",
  "arrayProperties": {
    "size": 7
  },
  "jobDefinition": "print-color"
```

```
}
```

2. 将任务提交到您的 AWS Batch 任务队列。记下在输出中返回的作业 ID。

```
$ aws batch submit-job --cli-input-json file://print-color-job.json
```

3. 描述作业的状态并等待作业移动到 SUCCEEDED。

步骤 5：查看数组作业日志

任务达到SUCCEEDED状态后，您可以从任务的容器中查看 CloudWatch 日志。

要在 “日志” 中查看作业的 CloudWatch 日志

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 在左侧导航窗格中，选择 Jobs (作业)。
3. 对于 Job queue(作业队列)，选择一个队列。
4. 在 Status (状态) 部分中，选择 succeeded (已成功)。
5. 要显示数组作业的所有子作业，请选择在上一节中返回的作业 ID。
6. 要从作业的容器查看日志，请选择其中一个子作业，然后选择 View logs (查看日志)。

7. 查看其他子作业的日志。每个作业将分别返回彩虹的一种颜色。

多节点并行作业

利用多节点并行作业，您能够跨多个 Amazon EC2 实例运行单个作业。借助 AWS Batch 多节点并行作业，您可以运行大规模、高性能计算应用程序和分布式 GPU 模型训练，而无需直接启动、配置和管理 Amazon EC2 资源。 AWS Batch 多节点 parallel 作业与任何支持基于 IP 的节点间通信的框架兼容。示例包括 Apache MxNet TensorFlow、、Caffe2 或消息传递接口 (MPI)。

多节点并行作业可作为单个作业提交。不过，作业定义（或作业提交节点覆盖）指定了要为作业创建的节点数量，以及要创建的节点组。每个多节点并行作业都包含一个主节点，这是最先启动的节点。主节点启动之后，子节点会启动并开始运行。只有当主节点退出时，作业才会完成。然后停止所有子节点。有关更多信息，请参阅 节点组。

多节点并行作业节点为单租户。这意味着，在每个 Amazon ECS 实例上只运行一个作业容器。

最终的作业状态（SUCCEEDED 或 FAILED）由主节点的最终作业状态决定。要获取多节点并行作业的状态，可以使用提交作业时返回的作业 ID 来描述作业。如果需要子节点的详细信息，则必须分别描述每个子节点。您可以使用 *#N* 表示法（从 0 开头）对节点进行寻址。例如，要访问任务的第二个节点的详细信息，请使用 API 操作描述 *aws_batch_job_* id #1。 AWS Batch DescribeJobs多节点并行作业的 `started`、`stoppedAt`、`statusReason` 和 `exit` 信息从主节点进行填充。

如果指定作业重试次数，则主节点故障会导致再次尝试重试。子节点故障不会导致更多的尝试发生。多节点并行作业每次新的尝试都将更新其关联子节点的相应尝试。

要在上运行多节点 parallel 作业 AWS Batch，您的应用程序代码必须包含分布式通信所需的框架和库。

环境变量

在运行时，每个节点都配置了所有 AWS Batch 作业都会收到的标准环境变量。此外，还配置了以下环境变量，这些变量特定于多节点并行作业：

`AWS_BATCH_JOB_MAIN_NODE_INDEX`

此变量设置为作业的主节点的索引号。您的应用程序代码可以将 `AWS_BATCH_JOB_MAIN_NODE_INDEX` 与单个节点上的 `AWS_BATCH_JOB_NODE_INDEX` 进行比较，以确定它是否为主节点。

`AWS_BATCH_JOB_MAIN_NODE_PRIVATE_IPV4_ADDRESS`

此变量仅在多节点并行作业子节点中设置。主节点上不存在此变量。此变量设置为作业的主节点的私有 IPv4 地址。您的子节点的应用程序代码可以使用此地址与主节点进行通信。

`AWS_BATCH_JOB_NODE_INDEX`

此变量设置为节点的节点索引号。节点索引从 0 开始，并且每个节点接收一个唯一的索引号。例如，包含 10 个子级的多节点并行作业具有索引值 0-9。

`AWS_BATCH_JOB_NUM_NODES`

此变量设置为您为多节点并行作业请求的节点数。

节点组

节点组即共享相同容器属性的一组完全相同的作业节点。您可以使用 AWS Batch 为每个作业指定最多五个不同的节点组。

每个组都有自己的容器映像、命令、环境变量等。例如，您可以提交一项作业，要求有一个主节点 `c5.xlarge` 实例和五个 `c5.xlarge` 实例子节点。这些不同的节点组中的每一个都可以为每个作业指定不同的容器映像或命令来运行。

或者，作业中的所有节点都可以使用单个节点组。此外，您的应用程序代码可以区分节点角色，例如主节点和子节点。它通过将 `AWS_BATCH_JOB_MAIN_NODE_INDEX` 环境变量与其自身的 `AWS_BATCH_JOB_NODE_INDEX` 值进行比较来实现此目的。单个作业中最多可具有 1000 个节点。这是 Amazon ECS 群集中的实例数的默认限制。您可以 请求增加此限制的值。

> **Note**
>
> 目前，多节点并行作业中的所有节点组必须使用相同的实例类型。

作业生命周期

当您提交多节点并行作业时，该作业会进入 `SUBMITTED` 状态。然后，该作业会等待任何作业依赖关系完成。作业也会变为 `RUNNABLE` 状态。最后 AWS Batch ，配置运行任务所需的实例容量并启动这些实例。

每个多节点并行作业都包含一个主节点。主节点是单个子任务， AWS Batch 用于监控以确定已提交的多节点作业的结果。主节点将第一个启动，并进入 `STARTING` 状态。`attemptDurationSeconds` 参数中指定的超时值适用于整个作业，而不适用于节点。

当主节点达到 `RUNNING` 状态时（节点的容器运行之后），子节点将启动，也进入 `STARTING` 状态。子节点的启动顺序是随机的。子节点启动的时机或顺序无法保证。要确保作业的所有节点都处于 `RUNNING` 状态（节点的容器运行之后），应用程序代码可以查询 AWS Batch API 来获取主节点和子节点信息。或者，应用程序代码可以等到所有节点都联机后再开始任何分布式处理任务。主节点的私有 IP 地址可作为 `AWS_BATCH_JOB_MAIN_NODE_PRIVATE_IPV4_ADDRESS` 环境变量，用在每个子节点中。应用程序代码可以使用此信息，在每个任务之间协调和传输数据。

各个节点在退出后，将进入 `SUCCEEDED` 或 `FAILED` 状态，具体取决于其退出代码。如果主节点退出，则作业将视为已完成，并且所有子节点将停止。如果子节点死亡，则 AWS Batch 不会对任务中的其他节点执行任何操作。如果节点数量减少后不希望作业继续，那么必须在应用程序代码中考虑此因素。这样做会终止或取消作业。

计算环境注意事项

在配置使用 AWS Batch运行多节点并行作业的计算环境时，需要考虑几个注意事项。

- UNMANAGED 计算环境不支持多节点并行作业。
- 如果打算将多节点并行作业提交到计算环境，请在单个可用区中创建集群 置放群组，并将其与计算资源进行关联。这样可保证多节点并行作业位于实例的逻辑分组内，同时保持高的网络流量潜力。有关更多信息，请参阅《Amazon EC2 用户指南》中的置放群组。
- 使用竞价型实例的计算环境不支持多节点并行作业。
- AWS Batch 多节点并行任务使用 Amazon ECS awsvpc 网络模式，该模式为您的多节点并行任务容器提供了与 Amazon EC2 实例相同的联网属性。每个多节点并行作业容器都可获得自己的弹性网络接口、主要私有 IP 地址以及内部 DNS 主机名。在同一 VPC 子网中创建网络接口，作为其主机计算资源。适用于计算资源的任何安全组，也适用于该主机计算资源。有关更多信息，请参阅 《Amazon Elastic Container Service 开发人员指南》 中的 使用 awsvpc Network Mode 的任务联网。
- 与计算环境关联的安全组不得超过 5 个。
- 对于具有公有 IP 地址的多节点并行作业，awsvpc 网络模式不提供弹性网络接口。要访问 Internet，必须在配置为使用 NAT 网关的私有子网中启动计算资源。有关更多信息，请参阅 Amazon VPC 用户指南中的 NAT 网关。节点间通信必须使用节点的私有 IP 地址或 DNS 主机名。运行于公有子网内计算资源之上的多节点并行作业，无出站网络访问。要创建具有私有子网和 NAT 网关的 VPC，请参阅创建虚拟私有云 。
- 创建并附加到计算资源的弹性网络接口，不能由您的账户手动分离或修改。这是为了防止意外删除与正在运行的作业关联的弹性网络接口。要释放任务的弹性网络接口，请终止作业。
- 计算环境必须具有足够大的 vCPU 才能支持多节点并行作业。
- 您的 Amazon EC2 实例限额包括运行作业所需的实例数量。例如，如果作业需要 30 个实例，但您的账户在区域内只能运行 20 个实例。则您的工作会卡在 RUNNABLE 状态中。
- 如果为多节点并行作业中的节点组指定了实例类型，那么计算环境必须能够启动该实例类型。

GPU 作业

借助 GPU 作业，您可以运行使用实例 GPU 的作业。

支持以下基于 GPU 的 Amazon EC2 实例类型。有关更多信息，请参阅 Amazon EC2 G3 实例、Amazon EC2 G4 实例、Amazon EC2 G5 实例、Amazon EC2 P2 实例、Amazon EC2 P3 实例、Amazon EC2 P4d 实例 和 Amazon EC2 P5 实例。

实例类型	GPU	GPU 内存	vCPU	内存	网络带宽
g3s.xlarge	1	8 GiB	4	30.5 GiB	10 Gbps
g3.4xlarge	1	8 GiB	16	122 GiB	最高 10 Gbps
g3.8xlarge	2	16 GiB	32	244 GiB	10 Gbps
g3.16xlarge	4	32 GiB	64	488 GiB	25 Gbps
g4dn.xlarge	1	16 GiB	4	16 GiB	最高 25 Gbps
g4dn.2xlarge	1	16 GiB	8	32 GiB	最高 25 Gbps
g4dn.4xlarge	1	16 GiB	16	64 GiB	最高 25 Gbps
g4dn.8xlarge	1	16 GiB	32	128 GiB	50 Gbps
g4dn.12xlarge	4	64 GiB	48	192 GiB	50 Gbps
g4dn.16xlarge	1	16 GiB	64	256 GiB	50 Gbps
g5.xlarge	1	24 GiB	4	16 GiB	最高 10 Gbps
g5.2xlarge	1	24 GiB	8	32 GiB	最高 10 Gbps
g5.4xlarge	1	24 GiB	16	64 GiB	最高 25 Gbps
g5.8xlarge	1	24 GiB	32	128 GiB	25 Gbps
g5.16xlarge	1	24 GiB	64	256 GiB	25 Gbps
g5.12xlarge	4	96 GiB	48	192 GiB	40Gbps
g5.24xlarge	4	96 GiB	96	384 GiB	50 Gbps
g5.48xlarge	8	192 GiB	192	768 GiB	100 Gbps
p2.xlarge	1	12 GiB	4	61 GiB	高
p2.8xlarge	8	96 GiB	32	488 GiB	10 Gbps

实例类型	GPU	GPU 内存	vCPU	内存	网络带宽
p2.16xlarge	16	192 GiB	64	732 GiB	20 Gbps
p3.2xlarge	1	16 GiB	8	61 GiB	最高 10 Gbps
p3.8xlarge	4	64 GiB	32	244 GiB	10 Gbps
p3.16xlarge	8	128 GiB	64	488 GiB	25 Gbps
p3dn.24xlarge	8	256 GiB	96	768 GiB	100 Gbps
p4d.24xlarge	8	320 GiB	96	1152 GiB	4x100 Gbps
p5.48xlarge	8	640 GiB	192	2 TiB	32x100 Gbps

> **Note**
>
> AWS Batch中的 GPU 作业仅支持支持 NVIDIA GPU 且使用 x86_64 架构的实例类型。例如，不支持 G4ad 和G5g 实例系列。

作业定义的 resourceRequirements 参数指定要固定到容器的 GPU 的数量。在作业持续期间，在该实例上运行的任何其他作业都无法使用此数量的 GPU。计算环境中将运行 GPU 作业的所有实例类型都应来自 p2、p3、p4、p5、g3、g3s、g4 或 g5 实例系列。如果不这么做，GPU 作业可能会停滞于 RUNNABLE 状态。

不使用 GPU 的作业可以在 GPU 实例上运行。但是，在 GPU 实例上运行它们的成本可能高于在类似的非 GPU 实例上运行的成本。根据具体的 vCPU、内存和所需时间，这些非 GPU 作业可能会阻止 GPU 作业运行。

在 Amazon EKS 资源上创建基于 GPU 的任务

本节介绍如何在 AWS Batch上运行 Amazon EKS GPU 工作负载。

目录

- 在您的 Amazon EKS 集群中运行 GPU 作业

在 Amazon EKS 上创建基于 GPU 的 Kubernetes 集群

在 Amazon EKS 上创建基于 GPU 的 Kubernetes 集群之前，您必须已完成 亚马逊 EK AWS Batch S 入门 中的步骤。此外，请考虑以下操作：

- AWS Batch 支持采用 NVIDIA GPU 的实例类型。
- 默认情况下， AWS Batch 选择版本与您的 Amazon EKS 集群控制平面Kubernetes版本匹配的 Amazon EKS 加速 AMI。

```
$ cat <<EOF > ./batch-eks-gpu-ce.json
{
  "computeEnvironmentName": "My-Eks-GPU-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:<region>:<account>:cluster/<cluster-name>",
    "kubernetesNamespace": "my-aws-batch-namespace"
  },
  "computeResources": {
    "type": "EC2",
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "minvCpus": 0,
    "maxvCpus": 1024,
    "instanceTypes": [
      "p3dn.24xlarge",
      "p4d.24xlarge"
    ],
    "subnets": [
        "<eks-cluster-subnets-with-access-to-internet-for-image-pull>"
    ],
    "securityGroupIds": [
        "<eks-cluster-sg>"
    ],
    "instanceRole": "<eks-instance-profile>"
  }
}
EOF
```

```
$ aws batch create-compute-environment --cli-input-json file://./batch-eks-gpu-ce.json
```

AWS Batch 不代表您管理 NVIDIA GPU 设备插件。您必须将此插件安装到您的 Amazon EKS 集群中，并允许它以 AWS Batch 节点为目标。有关更多信息，请参阅上的 “启用 GPU Suppor Kubernetes t” GitHub。

要将NVIDIA设备插件 (DaemonSet) 配置为以 AWS Batch 节点为目标，请运行以下命令。

```
# pull nvidia daemonset spec
$ curl -O https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.12.2/nvidia-device-plugin.yml
# using your favorite editor, add Batch node toleration
# this will allow the DaemonSet to run on Batch nodes
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"

$ kubectl apply -f nvidia-device-plugin.yml
```

我们不建议您将基于计算（CPU 和内存）的工作负载与基于 GPU 的工作负载混合在计算环境和作业队列的相同配对中。这是因为计算作业可能会耗尽 GPU 容量。

要连接作业队列，请运行以下命令。

```
$ cat <<EOF > ./batch-eks-gpu-jq.json
 {
    "jobQueueName": "My-Eks-GPU-JQ1",
    "priority": 10,
    "computeEnvironmentOrder": [
      {
        "order": 1,
        "computeEnvironment": "My-Eks-GPU-CE1"
      }
    ]
  }
EOF

$ aws batch create-job-queue --cli-input-json file://./batch-eks-gpu-jq.json
```

创建 Amazon EKS GPU 任务定义

目前仅支持 nvidia.com/gpu，并且您设置的资源值必须为整数。不能使用 GPU 片段。有关更多信息，请参阅 Kubernetes 文档 中的 Schedule GPUs。

要为 Amazon EKS 注册 GPU 作业定义，请运行以下命令。

```
$ cat <<EOF > ./batch-eks-gpu-jd.json
{
    "jobDefinitionName": "MyGPUJobOnEks_Smi",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "hostNetwork": true,
            "containers": [
                {
                    "image": "nvcr.io/nvidia/cuda:10.2-runtime-centos7",
                    "command": ["nvidia-smi"],
                    "resources": {
                        "limits": {
                            "cpu": "1",
                            "memory": "1024Mi",
                            "nvidia.com/gpu": "1"
                        }
                    }
                }
            ]
        }
    }
}
EOF

$ aws batch register-job-definition --cli-input-json file://./batch-eks-gpu-jd.json
```

在您的 Amazon EKS 集群中运行 GPU 作业

GPU 资源不可压缩。 AWS Batch 为请求值等于限制值的 GPU 作业创建 Pod 规范。这是一项Kubernetes要求。

要提交 GPU 作业，请运行以下命令。

```
$ aws batch submit-job --job-queue My-Eks-GPU-JQ1 --job-definition MyGPUJobOnEks_Smi --
job-name My-Eks-GPU-Job
```

```
# locate information that can help debug or find logs (if using Amazon CloudWatch Logs
 with Fluent Bit)
$ aws batch describe-jobs --job <job-id> | jq '.jobs[].eksProperties.podProperties |
 {podName, nodeName}'
{
  "podName": "aws-batch.f3d697c4-3bb5-3955-aa6c-977fcf1cb0ca",
  "nodeName": "ip-192-168-59-101.ec2.internal"
}
```

搜索和筛选 AWS Batch 职位

您可以使用 AWS Batch 控制台列出任务队列中的作业。但是，如果作业队列中有许多作业，则可能很难找到特定的作业。

您可以使用搜索和筛选来列出与您指定的搜索条件相匹配的作业。

1. 打开AWS Batch 控制台。
2. 选择 Jobs (作业)。
3. 开启搜索和筛选。

> **Note**
>
> 如果您有多份作业，此过程可能需要几分钟。

4. 在请选择作业队列框中，选择您要搜索的作业队列。
5. 在按属性或值筛选资源框中，选择列出的属性之一。
6. 选择要使用的运算符。例如，选择 Status =。

> **Tip**
>
> 要使用其他属性或运算符，请关闭当前条件。然后，选择所需的属性和运算符。

7. 输入或选择属性值。例如，输入全部或部分作业名称或选择 Status = RUNNABLE。
8. 在筛选列表中选择所需的作业。

> **Tip**
>
> 如果看不到所需的作业，请滚动已筛选的列表。

作业日志

您可以将 AWS Batch 作业配置为将日志信息发送到 CloudWatch 日志。这样，您可以在一个方便的位置查看作业的不同日志。有关更多信息，请参阅 将 CloudWatch 日志与配合使用 AWS Batch。

您还可以使用 AWS Batch 控制台中的 Job 日志来监控作业或对 AWS Batch 作业进行故障排除。

1. 打开AWS Batch 控制台。
2. 选择 Jobs (作业)。
3. 对于作业队列，选择所需的作业队列。

> **Tip**
>
> 如果作业队列中有多个作业，则可以打开搜索和筛选以更快地找到作业。有关更多信息，请参阅 搜索和筛选 AWS Batch 职位。

4. 在状态 中，选择所需的作业状态。
5. 选择所需的作业。
6. 在详细信息页面上，向下滚动到作业日志。
7. 选择检索日志。
8. 在 “需要授权” 中，输入**OK**，然后选择 “授权” 以接受 Amazon 的 CloudWatch 费用。

> **Note**
>
> 要撤销您的 CloudWatch 收费授权，请执行以下操作：
>
> 1. 在左侧导航窗格中，请选择权限。
> 2. 对于作业日志，选择编辑。
> 3. 清除 “授权 Batch 使用” CloudWatch 复选框。
> 4. 选择保存更改。

9. 查看 AWS Batch 作业的日志数据。

> **Tip**
>
> 您可以根据关键字、最大结果数和排序来筛选日志。您也可以选择一个默认时间间隔或创建自定义间隔来自定义结果。

作业信息

您可以查看 AWS Batch 作业信息，例如状态、作业定义和容器信息。

1. 打开AWS Batch 控制台。
2. 选择 Jobs (作业)。
3. 对于作业队列，选择所需的作业队列。

> **Tip**
>
> 如果作业队列中有多个作业，则可以打开搜索和筛选以更快地找到作业。有关更多信息，请参阅 搜索和筛选 AWS Batch 职位。

4. 选择所需的作业。

> **Note**
>
> 您还可以使用 AWS Command Line Interface (AWS CLI) 来查看有关 AWS Batch 作业的详细信息。有关更多信息，请参阅 AWS CLI 命令引用中的 describe-jobs。

作业定义

AWS Batch 作业定义指定作业的运行方式。虽然每个作业必须引用作业定义，但可在运行时覆盖作业定义中指定的许多参数。

内容

在作业定义中指定的一些属性包括：

- 在作业中用于容器的 Docker 映像
- 要将多少个 vCPU 和多少内存量用于容器
- 容器在启动时应运行的命令
- 当容器启动时，应传递给容器的环境变量（如果有）
- 应该用于容器的任何数据卷
- 您的任务应使用哪个（如果有）IAM 角色来获得 AWS 权限

有关作业定义中可用的参数的完整描述，请参阅的 Job 定义参数 ContainerProperties。

创建单节点作业定义

您可在 AWS Batch 中运行作业之前，必须先创建一个作业定义。对于单节点并行作业和多节点并行作业，此过程略有不同。本主题专门介绍如何为不是多节点并行作业的 AWS Batch 作业创建作业定义。

您可以在 Amazon 弹性容器服务资源上创建多节点并行作业定义。有关更多信息，请参阅 the section called “创建多节点并行作业定义”。

主题

在 Amazon EC2 资源上创建单节点作业定义

要在 Amazon EC2 资源上创建新的作业定义，请执行以下操作：

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的 AWS 区域。
3. 在左侧导航窗格中，选择作业定义。
4. 选择 Create（创建）。
5. 对于编排类型，选择 Amazon Elastic Compute Cloud (Amazon EC2)。
6. 对于 EC2 平台配置，请关闭启用多节点并行处理。
7. 对于名称，为您的作业定义输入唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
8. （可选）对于执行超时，输入超时值（以秒为单位）。执行超时是指未完成的作业终止之前的时间长度。如果某次尝试超过了超时时间，该尝试将停止，状态将转为 FAILED。有关更多信息，请参阅作业超时。最小值为 60 秒。
9. （可选）开启计划优先级。输入介于 0 到 100 之间的计划优先级值。值越高，优先级越高。
10. （可选）对于作业尝试，请输入 AWS Batch 尝试将作业移至 RUNNABLE 状态的次数。请输入 1 到 10 之间的数字。
11. （可选）对于重试策略条件，选择退出时添加评估。至少输入一个参数值，然后选择一个操作。对于每组条件，必须将操作设置为重试或退出。这些操作意味着以下几点：

    - 重试 – AWS Batch 重试，直到达到您指定的作业尝试次数。
    - 退出 – AWS Batch 停止重试作业。

> **Important**
>
> 如果选择退出时添加评估，则必须至少配置一个参数并选择一个操作或选择退出时移除评估。

12. （可选）展开 标签，然后选择添加标签以向资源添加标签。输入键和可选的值，然后选择添加标签。
13. （可选）开启 传播标签将标签从作业和作业定义传播到 Amazon ECS 任务。
14. 选择下一页。
15. 在容器配置部分：

    a. 对于映像，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。您也可以使用 *repository-url*/*image*:*tag* 指定其他存储库。名称长度不超过 225 个字符。可以包含大小写字母、数字、连字符（-）、下划线（_）、冒号（:）、正斜杠（/）和数字符号（#）。此参数可映射到 Docker Remote API 的创建容器部分中的 Image 和 docker run 的 IMAGE 参数。

    > **Note**
    >
    > Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于 Arm 的 Docker 映像只能在基于 Arm 的计算资源上运行。

    - Amazon ECR 公有存储库中的映像使用完整的`registry/repository[:tag]`或`registry/repository[@digest]`命名惯例（例如，`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*）。
    - Amazon ECR 存储库中的映像使用完整的`registry/repository[:tag]`命名惯例（例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*）。
    - Docker Hub 上的官方存储库中的映像使用一个名称（例如，`ubuntu` 或 `mongo`）。
    - Docker Hub 上其他存储库中的映像通过组织名称（例如，`amazon/amazon-ecs-agent`）进行限定。
    - 其他在线存储库中的映像由域名 (例如，`quay.io/assemblyline/ubuntu`) 进行进一步限定。

    b. 对于命令语法，请选择 Bash 或 JSON。

    c. 对于 Command，指定要传递到容器的命令。对于更简单的命令，请像输入命令提示符一样输入命令。然后，验证 JSON 结果是否正确，且是否传递给 Docker daemon。对于更复杂的命令（例如，使用特殊字符），请使用 JSON 语法。

> **Tip**
>
> 选择信息以查看 Bash 和 JSON 编码示例。

此参数映射到 Docker Remote API 创建容器部分中的 `Cmd`，以及 docker run 的 `COMMAND` 参数。有关 Docker `CMD` 参数的更多信息，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。

> **Note**
>
> 您可以在命令中使用参数替代默认值和占位符。有关更多信息，请参阅参数。

d. （可选）对于执行角色，指定一个 IAM 角色，该角色授予 Amazon ECS 容器代理代表您进行 AWS API 调用的权限。此功能使用 Amazon ECS IAM 角色执行任务。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 任务执行 IAM 角色。

e. 在作业角色配置中，请选择有权访问 AWS API 的 IAM 角色。此功能使用 Amazon ECS IAM 角色执行任务。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的任务的 IAM 角色。

> **Note**
>
> 此处仅显示具有 Amazon Elastic Container Service Task Role 信任关系的角色。有关为 AWS Batch 作业创建 IAM 角色的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的为任务创建 IAM 角色和策略。

16. 对于参数，选择添加参数以添加参数替换占位符作为键和可选值对。
17. 在环境配置部分：

    a. 对于 vCPUs，输入要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的 `CpuShares` 以及 docker run 的 `--cpu-shares` 选项。每个 vCPU 相当于 1024 个 CPU 份额。您必须指定至少一个 vCPU。

    b. 对于内存，输入容器可用的内存限制。如果您的容器尝试使用超出您在此处指定的内存量，该容器将会被终止。此参数将映射到 Docker Remote API 的创建容器部分中的 `Memory` 以及 docker run 的 `--memory` 选项。您必须为作业指定至少 4 MiB 内存。

> **Note**
>
> 要最大限度地提高资源利用率，请为特定实例类型的作业确定内存优先级。有关更多信息，请参阅计算资源内存管理。

c. 在 GPU 数量中，选择要为容器预留的 GPU 数量。

d. （可选）对于环境变量，选择添加环境变量以名称-值对的形式添加环境变量。这些变量传递给容器。

e. （可选）对于密钥，选择添加密钥，将密钥添加为名称-值对。这些密钥暴露在容器中。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

18. 选择下一页。

19. 在 Linux 配置部分中：

a. 对于 User，输入要在容器内使用的用户名。此参数将映射到 Docker Remote API 的创建容器部分中的 `User` 以及 docker run 的 `--user` 选项。

b. （可选）要授予作业容器对主机实例（类似于 `root` 用户）的更高权限，请向右拖动权限滑块。此参数将映射到 Docker Remote API 的创建容器部分中的 `Privileged` 以及 docker run 的 `--privileged` 选项。

c. （可选）开启启用 Init 处理以在容器内运行 `init` 进程。该进程转发信号和获得进程。

20. （可选）在文件系统配置部分：

a. 开启启用只读文件系统以移除对卷的写入权限。

b. 在共享内存大小中，输入 `/dev/shm` 卷的大小（以 MiB 为单位）。

c. 在最大交换大小中，输入容器可使用的总交换内存量（以 MiB 为单位）。

d. 在 Swappiness 中输入一个介于 0 和 100 之间的值，以指示容器的 swappiness 行为。如果不指定值且启用了交换，则值默认值为 60。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 swappiness。

e. （可选）展开其他配置。

f. （可选）对于 Tmpfs，请选择添加 tmpfs 以添加 `tmpfs` 挂载。

g. （可选）对于设备，选择添加设备以添加设备：

i. 对于容器路径，指定容器实例中的路径以公开映射到主机实例的设备。如果将其留空，则在容器中使用主机路径。

ii. 对于主机路径，指定主机实例中设备的路径。

iii. 对于权限，选择要应用于设备的一个或多个权限。可用权限包括读取、写入和 MKNOD。

h. （可选）对于卷配置，请选择添加卷以创建要传递到容器的卷列表。输入卷的名称和源路径，然后选择添加卷。您也可以选择开启启用 EFS。

i. （可选）对于挂载点，请选择添加挂载点配置以添加数据卷的挂载点。您必须指定源卷和容器路径。这些挂载点会传递到容器实例上的 Docker daemon。您也可以选择将卷设为只读。

j. （可选）对于 Ulimits 配置，请选择添加 ulimit 为容器添加一个 `ulimits` 值。输入名称、软限制和硬限制值，然后选择添加 ulimit。

21. （可选）在日志记录配置部分：

a. 对于日志驱动程序，请选择要使用的日志驱动程序。有关可用日志驱动程序的更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 logDriver。

> **Note**
>
> 默认情况下，使用 `awslogs` 日志驱动程序。

b. 在选项中，选择添加选项以添加选项。输入名称-值对，然后选择添加选项。

c. 对于密钥，选择添加密钥。输入名称-值对，然后选择添加密钥以添加密钥。

> **Tip**
>
> 有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

22. 选择下一页。

23. 对于作业定义查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建作业定义。

在 AWS Fargate 资源上创建单节点作业定义

有关 AWS Fargate 资源的新作业定义，请执行以下操作：

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 从顶部导航栏中，选择要使用的 AWS 区域。
3. 在左侧导航窗格中，选择作业定义。
4. 选择创建。
5. 对于编排类型，请选择 Fargate。有关更多信息，请参见 AWS Batch 在 AWS Fargate。

6. 对于名称，为您的作业定义输入唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
7. （可选）对于执行超时，输入超时值（以秒为单位）。执行超时是指未完成的作业终止之前的时间长度。如果某次尝试超过了超时时间，该尝试将停止，状态将转为 FAILED。有关更多信息，请参见 作业超时。最小值为 60 秒。
8. （可选）开启计划优先级。输入介于 0 到 100 之间的计划优先级值。值越高，相较于较低值的优先级越高。
9. （可选）展开 标签，然后选择添加标签以向资源添加标签。启用传播标签以传播作业和作业定义中的标签。
10. 在 Fargate 平台配置部分中：
    a. 对于运行时平台，请选择计算环境架构。
    b. 对于操作系统系列，为计算环境选择操作系统。
    c. 对于 CPU 架构，请选择 vCPU 架构。
    d. 对于 Fargate 平台版本，请输入 LATEST 或特定的运行时系统环境版本。
    e. （可选）打开分配公有 IP，为 Fargate 作业网络接口分配公有 IP 地址。为使在私有子网中运行的作业将出站流量发送到互联网，私有子网需要挂载 NAT 网关才能将请求路由到互联网。您可能需要这样做，以便可以提取容器映像。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 任务联网。
    f. （可选）对于临时存储，请输入要分配给任务的临时存储量。临时存储容量必须介于 21 GiB 到 200 GiB 之间。默认情况下，如果您不输入值，则会分配 20 GiB 的临时存储空间。

       > **Note**
       > 临时存储需要 Fargate 平台版本 1.4 或更高版本。

    g. 对于 执行角色，指定 IAM 角色，该角色授予 Amazon ECS 容器和 Fargate 代理代表您进行 AWS API 调用的权限。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南 中的 Amazon ECS 任务执行 IAM 角色。
    h. 对于作业尝试，请输入 AWS Batch 尝试将作业移至某一 RUNNABLE 状态的次数。请输入 1 到 10 之间的数字。
    i. 可选）对于重试策略条件，选择退出时添加评估。至少输入一个参数值，然后选择一个操作。对于每组条件，必须将操作设置为重试或退出。这些操作意味着以下几点：

- 重试 — AWS Batch 重试，直到达到您指定的作业尝试次数。
- 退出 – AWS Batch 停止重试作业。

> **Important**
>
> 如果选择退出时添加评估，则必须至少配置一个参数并选择一个操作或选择退出时移除评估。

11. 选择下一页。
12. 在容器配置部分：

    a. 对于 映像，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。也可以使用*repository-url*/*image*:*tag*指定其他存储库。名称长度不超过 225 个字符。可以包含大小写字母、数字、连字符（-）、下划线（_）、冒号（:）、句点（.）、正斜杠（/）和数字符号（#）。此参数可映射到 Docker Remote API 的创建容器部分中的Image和docker run的IMAGE参数。

    > **Note**
    >
    > Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于Arm的Docker映像只能在基于Arm的计算资源上运行。

    - Amazon ECR 公有存储库中的映像使用完整的registry/repository[:tag]或registry/repository[@digest]命名惯例（例如，public.ecr.aws/*registry_alias*/*my-web-app*:*latest*）。
    - Amazon ECR 存储库中的映像使用完整的registry/repository[:tag]命名惯例（例如，*aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest*）。
    - Docker Hub 上的官方存储库中的映像使用单一名称 (例如，ubuntu 或 mongo)。
    - Docker Hub 上其他存储库中的映像通过组织名称（例如，amazon/amazon-ecs-agent）进行限定。
    - 其他在线存储库中的映像由域名 (例如，quay.io/assemblyline/ubuntu) 进行进一步限定。

    b. 对于命令语法，请选择 Bash 或 JSON。

c. 对于 Command，指定要传递到容器的命令。对于简单的命令，请像输入命令提示符一样输入命令，然后验证 JSON 结果是否正确。它传递给 Docker 进程守护程序。对于更复杂的命令（例如，使用特殊字符），请使用 JSON 语法。

> **Tip**
>
> 选择信息以查看 Bash 和 JSON 编码示例。

此参数映射到 Docker Remote API 创建容器部分中的Cmd，以及docker run的COMMAND参数。有关 Docker CMD 参数的更多信息，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。

> **Note**
>
> 您可以在命令中使用参数替代默认值和占位符。有关更多信息，请参见 参数。

d. （可选）将参数作为名称-值映射添加到作业定义中，以覆盖作业定义的默认值。若要添加参数：

- 对于参数，选择添加参数，输入名称-值对，然后选择添加参数。

> **Important**
>
> 如果选择添加参数，则必须至少配置一个参数或选择移除参数

e. 在环境配置部分：

i. 在作业角色配置中，请选择有权访问 AWS API 的 IAM 角色。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的任务的 IAM 角色。

> **Note**
>
> 此处仅显示具有 Amazon Elastic Container Service Task Role 信任关系的角色。有关为 AWS Batch 作业创建 IAM 角色的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的为任务创建 IAM 角色和策略。

ii. 对于 vCPUs，输入要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的CpuShares以及docker run的--cpu-shares选项。每个 vCPU 相当于 1024 个 CPU 份额。您必须指定至少一个 vCPU。

iii. 对于内存，输入容器可用的内存限制。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的创建容器部分中的Memory以及docker run的--memory选项。您必须为作业指定至少 4 MiB 内存。

如果您使用 GuardDuty 运行时监控，则 GuardDuty 安全代理会有轻微的内存开销。因此，内存限制必须包括 GuardDuty安全代理的大小。有关 GuardDuty 安全代理内存限制的信息，请参阅《GuardDuty 用户指南》中的 CPU 和内存限制。有关最佳实践的信息，请参阅 Amazon ECS 开发人员指南中的启用运行时监控后如何修复 Fargate 任务中的内存不足错误。

> **Note**
>
> 要最大限度地提高资源利用率，请为特定实例类型的作业设置内存优先级。有关更多信息，请参见 计算资源内存管理。

f. （可选）对于环境变量，选择添加环境变量以名称-值对的形式添加环境变量。这些变量传递给容器。

g. （可选）对于密钥，选择添加密钥，将密钥添加为名称-值对。这些密钥暴露在容器中。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

h. 选择下一页。

13. （可选）在 Linux 配置部分中：

a. 对于 用户，输入要在容器内使用的用户名。

b. 开启启用初始化进程以在容器内运行初始化进程。该进程转发信号和获得进程。

c. 开启启用只读文件系统以移除对卷的写入权限。

d. （可选）展开其他配置。

e. 对于挂载点配置，请选择添加挂载点配置以添加数据卷的挂载点。您必须指定源卷和容器路径。这些挂载点会传递到容器实例上的 Docker daemon。

f. 对于卷配置，请选择添加卷以创建要传递到容器的卷列表。输入卷的名称和源路径，然后选择添加卷。

g. 在日志配置部分：

i. （可选）对于日志驱动程序，选择要使用的日志驱动程序。有关可用日志驱动程序的更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 logDriver。

> **Note**
>
> 默认情况下，使用 `awslogs` 日志驱动程序。

ii. （可选）在选项中，选择添加选项以添加选项。输入名称-值对，然后选择添加选项。

iii. （可选）对于密钥，选择添加密钥以添加密钥。然后，输入名称-值对，并选择添加密钥。

> **Tip**
>
> 有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

14. 选择下一页。
15. 对于作业定义查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建作业定义。

在 Amazon EKS 资源上创建单节点作业定义

要在 Amazon Elastic Kubernetes Service 资源上创建新的作业定义，请执行以下操作：

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从顶部导航栏中，选择要使用的 AWS 区域。
3. 在左侧导航窗格中，选择作业定义。
4. 选择 Create（创建）。
5. 对于编排类型，选择 Elastic Kubernetes Service (EKS)。
6. 对于名称，为您的作业定义输入唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
7. （可选）对于执行超时，输入超时值（以秒为单位）。执行超时是指未完成的作业终止之前的时间长度。如果某次尝试超过了超时时间，该尝试将停止，状态将转为 `FAILED`。有关更多信息，请参阅作业超时。最小值为 60 秒。

8. （可选）开启计划优先级。输入介于 0 到 100 之间的计划优先级值。值越高，相较于较低值的优先级越高。
9. （可选）展开 标签，然后选择添加标签以向资源添加标签。
10. 选择下一页。
11. 在 EKS pod 属性部分中：
    a. 在服务账户名称中，输入为在 pod 中运行的进程提供身份的账户。
    b. 打开主机网络以使用 Kubernetes pod 网络模型，并为传入的连接打开侦听端口。仅对传出通信关闭此设置。
    c. 对于 DNS 策略，选择以下选项之一：
       - 无值（空）- pod 忽略 Kubernetes 环境中的 DNS 设置。
       - 默认 — pod 从其运行的节点继承名称解析配置。

         > **Note**
         > 如果未指定 DNS 策略，则默认不是默认 DNS 策略。而是使用 ClusterFirst。

       - ClusterFirst — 任何与配置的集群域后缀不匹配的 DNS 查询都将转发到继承自节点的上游名称服务器。
       - ClusterFirstWithHostNet — 如果主机网络已开启，则使用。
    d. （可选）对于容器组（pod）标签，选择添加容器组（pod）标签，然后输入名称/值对。

       > **Important**
       > 容器组（pod）标签的前缀不能包含 `kubernetes.io/`、`k8s.io/` 或 `batch.amazonaws.com/`。

    e. 选择下一页。
    f. 在容器配置部分：
       i. 对于名称，输入容器的名称。该名称必须以字母或数字开头，并且最多可以包含 25 个字符。可以包含大小写字母、数字和连字符。
       ii. 对于映像，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。您也可以使用 *`repository-url/image:tag`* 指定其他存储库。名称最多可以有 255 个字符。可以包含大小写字母、数字、连字符（-）、下划线（_）、冒号

（:）、句点（.）、正斜杠（/）和数字符号（#）。此参数可映射到 Docker Remote API 的创建容器部分中的 Image 和 docker run 的 IMAGE 参数。

> **Note**
>
> Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于 Arm 的 Docker 映像只能在基于 Arm 的计算资源上运行。

- Amazon ECR 公有存储库中的映像使用完整的`registry/repository[:tag]`或`registry/repository[@digest]`命名惯例（例如，`public.ecr.aws/`*`registry_alias`*`/`*`my-web-app`*`:`*`latest`*）。
- Amazon ECR 存储库中的映像使用完整的`registry/repository[:tag]`命名惯例（例如，*`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*）。
- Docker Hub 上的官方存储库中的映像使用单一名称 (例如，ubuntu 或 mongo)。
- Docker Hub 上其他存储库中的映像通过组织名称（例如，`amazon/amazon-ecs-agent`）进行限定。
- 其他在线存储库中的映像由域名 (例如，`quay.io/assemblyline/ubuntu`) 进行进一步限定。

iii. （可选）对于映像提取策略，请选择何时提取映像。

iv. （可选）在命令中，输入要传递到容器的 Bash 或 JSON 命令。

v. （可选）在参数中输入要传递给容器的参数。如果未提供参数，则使用容器映像命令。

g. （可选）您可以将参数作为键值映射添加到作业定义中，以覆盖作业定义的默认值。若要添加参数：

- 在参数中，输入名称/值对，然后选择添加参数。

> **Important**
>
> 如果选择添加参数，则必须至少配置一个参数或选择移除参数。

h. 在环境配置部分：

i. 对于 vCPUs，输入要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的 CpuShares 以及 docker run 的 --cpu-shares 选项。每个 vCPU 相当于 1024 个 CPU 份额。您必须指定至少一个 vCPU。

ii. 对于内存，输入容器可用的内存限制。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的创建容器部分中的 Memory 以及 docker run 的 --memory 选项。您必须为作业指定至少 4 MiB 内存。

> **Note**
>
> 要最大限度地提高资源利用率，请为特定实例类型的作业确定内存优先级。有关更多信息，请参阅计算资源内存管理。

i. （可选）对于环境变量，选择添加环境变量以名称-值对的形式添加环境变量。这些变量传递给容器。

j. （可选）对于卷装载：

i. 选择添加卷装载。

ii. 输入名称，然后在装入卷的容器中输入装载路径。

iii. 选择只读可删除该卷的写入权限。

iv. 选择添加卷装载。

k. （可选）在以用户身份运行中，输入用户 ID 以运行容器进程。

> **Note**
>
> 映像中必须存在用户 ID，容器才能运行。

l. （可选）在分组运行中，输入组 ID 以运行容器进程。

> **Note**
>
> 映像中必须存在群组 ID，容器才能运行。

m. （可选）要为您的作业容器授予对主机实例的提升权限（类似于 root 用户），请选择 特权。此参数将映射到 Docker Remote API 的创建容器部分中的 Privileged 以及 docker run 的 --privileged 选项。

n. （可选）打开只读根文件系统以删除对根文件系统的写入权限。

o. （可选）启用以非根用户身份运行，以非根用户身份运行 pod 中的容器。

> **Note**
>
> 如果启用以非根用户身份运行，则 kubelet 会在运行时验证映像以查证该映像不是以 UID 0 运行的。

p. 选择下一页。

12. 对于作业定义查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建作业定义。

创建多节点并行作业定义

您可在 AWS Batch 中运行作业之前，必须先创建一个作业定义。对于单节点并行作业和多节点并行作业，此过程略有不同。本主题具体介绍如何为 AWS Batch 多节点并行作业创建作业定义。有关更多信息，请参阅多节点并行作业。

> **Note**
>
> AWS Fargate 不支持多节点并行作业。

在 Amazon EC2 资源上创建多节点并行作业定义

要创建单节点作业定义，请参阅创建单节点作业定义 。

在 Amazon Elastic Compute Cloud 资源上创建多节点并行作业定义：

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的 AWS 区域。
3. 在导航窗格中，选择作业定义。
4. 选择 Create（创建）。
5. 对于编排类型，选择 Amazon Elastic Compute Cloud (Amazon EC2)。
6. 对于启用多节点并行，请打开多节点并行。
7. 对于名称，为您的作业定义输入唯一名称。名称可以长达 128 个字符，并且可以包含大小写字母、数字、连字符 (-) 和下划线 (_)。

8. （可选）对于 执行超时，指定您希望作业尝试运行的最大秒数。如果某次尝试超过了超时时间，该尝试将停止，状态将转为 FAILED。有关更多信息，请参阅作业超时。
9. （可选）开启计划优先级。输入介于 0 到 100 之间的计划优先级值。值越高，相较于较低值的优先级越高。
10. （可选）对于作业尝试，请输入 AWS Batch 尝试将作业移至 RUNNABLE 状态的次数。请输入 1 到 10 之间的数字。
11. （可选）对于重试策略条件，选择退出时添加评估。至少输入一个参数值，然后选择一个操作。对于每组条件，必须将操作设置为重试或退出。这些操作意味着以下几点：

    - 重试 – AWS Batch 重试，直到达到您指定的作业尝试次数。
    - 退出 – AWS Batch 停止重试作业。

    > **Important**
    >
    > 如果选择退出时添加评估，则必须至少配置一个参数并选择一个操作或选择退出时移除评估。

12. （可选）展开标签，然后选择添加标签以向资源添加标签。输入键和可选的值，然后选择添加标签。（可选）您也可开启传播标签，将标签从作业和作业定义传播到 Amazon ECS 任务。
13. 选择下一页。
14. 对于 Number of nodes (节点数)，输入要用于作业的节点的总数。
15. 对于 Main node (主节点)，输入要用于主节点的节点索引。默认主节点索引为 0。
16. 对于实例类型，选择实例类型。

    > **Note**
    >
    > 您选择的实例类型适用于所有节点。

17. 对于参数，选择添加参数以添加参数替换占位符作为键和可选值对。
18. 在节点范围部分中：

    a. 选择添加节点范围。这将创建节点范围部分。

    b. 对于 Target nodes (目标节点)，使用 *range_start*:*range_end* 表示法为节点组指定范围。

对于您为作业指定的节点，您可以创建最多 5 个节点范围。节点范围使用节点的索引值，并且节点索引从 0 开始。确保最终节点组的范围结束索引值比您指定的节点数少一。例如，假设您指定了 10 个节点，并且想要使用单个节点组。然后，您的终止范围是 9。

c. 对于映像，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。您也可以使用 *repository-url*/*image*:*tag* 指定其他存储库。名称最多可以有 255 个字符。可以包含大小写字母、数字、连字符（-）、下划线（_）、冒号（:）、正斜杠（/）和数字符号（#）。此参数可映射到 Docker Remote API 的创建容器部分中的 Image 和 docker run 的 IMAGE 参数。

> **Note**
>
> Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于 Arm 的 Docker 映像只能在基于 Arm 的计算资源上运行。

- Amazon ECR 公有存储库中的映像使用完整的registry/repository[:tag]或registry/repository[@digest]命名惯例（例如，public.ecr.aws/*registry_alias*/*my-web-app*:*latest*）。
- Amazon ECR 存储库中的映像使用完整的registry/repository[:tag]命名惯例。例如 *aws_account_id*.dkr.ecr.*region*.amazonaws.com/*my-web-app*:*latest*
- Docker Hub 上的官方存储库中的映像使用一个名称（例如，ubuntu 或 mongo）。
- Docker Hub 上其他存储库中的映像通过组织名称（例如，amazon/amazon-ecs-agent）进行限定。
- 其他在线存储库中的映像由域名 (例如，quay.io/assemblyline/ubuntu) 进行进一步限定。

d. 对于命令语法，请选择 Bash 或 JSON。

e. 对于 Command，指定要传递到容器的命令。对于简单的命令，您可以在 Space delimited 选项卡上输入命令，就像在命令提示符中键入命令一样。然后，验证 JSON 结果是否正确。JSON 结果将传递给 Docker daemon。对于较复杂的命令 (例如，带有特殊字符)，您可以切换到 JSON 选项卡，然后在该选项卡中输入等效字符串数组。

此参数映射到 Docker Remote API 创建容器部分中的 Cmd，以及 docker run 的 COMMAND 参数。有关 Docker CMD 参数的更多信息，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。

> **Note**
>
> 您可以在命令中使用参数替代默认值和占位符。有关更多信息，请参阅参数。

f. 对于 vCPUs，指定要为容器预留的 vCPU 数量。此参数将映射到 Docker Remote API 的创建容器部分中的 `CpuShares` 以及 docker run 的 `--cpu-shares` 选项。每个 vCPU 相当于 1024 个 CPU 份额。您必须指定至少一个 vCPU。

g. 对于 Memory，指定要提供给作业容器的内存硬限制 (以 MiB 为单位)。如果您的容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的创建容器部分中的 `Memory` 以及 docker run 的 `--memory` 选项。您必须为作业指定至少 4 MiB 内存。

> **Note**
>
> 您可以尝试通过为作业提供尽可能多的用于特定实例类型的内存来最大程度地利用资源。有关更多信息，请参阅计算资源内存管理。

h. （可选）对于 GPU 数 ，指定您的作业将使用的 GPU 的数量。该作业将在固定有指定数量的 GPU 的容器上运行。

i. （可选）对于任务角色，您可以指定一个 IAM 角色，该角色为任务中的容器提供使用 AWS API 的权限。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息（包括配置先决条件），请参阅 Amazon Elastic Container Service 开发人员指南中的任务中的 IAM 角色。

> **Note**
>
> 对于在 Fargate 资源上运行的作业，需要作业角色。

> **Note**
>
> 此处仅显示具有 Amazon Elastic Container Service Task Role 信任关系的角色。有关为 AWS Batch 作业创建 IAM 角色的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的为任务创建 IAM 角色和策略。

j. （可选）对于执行角色，指定一个 IAM 角色，该角色授予 Amazon ECS 容器代理代表您进行 AWS API 调用的权限。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 任务执行 IAM 角色。

19. （可选）展开其他配置：

a. 对于环境变量，选择添加环境变量以名称-值对的形式添加环境变量。这些变量传递给容器。

b. 对于任务角色配置，您可以指定一个 IAM 角色，该角色为任务中的容器提供使用 AWS API 的权限。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息（包括配置先决条件），请参阅 Amazon Elastic Container Service 开发人员指南中的任务中的 IAM 角色。

> **Note**
>
> 对于在 Fargate 资源上运行的作业，需要作业角色。

> **Note**
>
> 此处仅显示具有 Amazon Elastic Container Service Task Role 信任关系的角色。有关为 AWS Batch 作业创建 IAM 角色的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的为任务创建 IAM 角色和策略。

c. 对于执行角色，指定一个 IAM 角色，该角色授予 Amazon ECS 容器代理代表您进行 AWS API 调用的权限。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 任务执行 IAM 角色。

20. 在安全配置部分：

a. （可选）要为您的作业容器授予对主机实例的提升权限（类似于 `root` 用户），请启用 特权。此参数将映射到 Docker Remote API 的创建容器部分中的 `Privileged` 以及 docker run 的 `--privileged` 选项。

b. （可选）对于用户名，输入要在容器内使用的用户名。此参数将映射到 Docker Remote API 的创建容器部分中的 `User` 以及 docker run 的 `--user` 选项。

c. （可选）对于密钥，选择添加密钥，将密钥添加为名称-值对。这些密钥暴露在容器中。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

21. 在 Linux 配置部分中：

a. 开启启用只读文件系统以移除对卷的写入权限。

b. （可选）开启启用 init 进程以在容器内运行 `init` 进程。该进程转发信号和获得进程。

c. 在共享内存大小中，输入 `/dev/shm` 卷的大小（以 MiB 为单位）。

d. 在最大交换大小中，输入容器可使用的总交换内存量（以 MiB 为单位）。

e. 在 Swappiness 中输入一个介于 0 和 100 之间的值，以指示容器的 swappiness 行为。如果不指定值且启用了交换，则值默认值为 60。有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 swappiness。

f. （可选）对于设备，选择添加设备以添加设备：

   i. 对于容器路径，指定容器实例中的路径以公开映射到主机实例的设备。如果将其留空，则在容器中使用主机路径。

   ii. 对于主机路径，指定主机实例中设备的路径。

   iii. 对于权限，选择要应用于设备的一个或多个权限。可用权限包括读取、写入和 MKNOD。

22. （可选）对于挂载点，请选择添加挂载点配置以添加数据卷的挂载点。您必须指定源卷和容器路径。这些挂载点会传递到容器实例上的 Docker 进程守护程序。您也可以选择将卷设为只读。

23. （可选）对于 Ulimits 配置，请选择添加 ulimit 为容器添加一个 `ulimits` 值。输入名称、软限制和硬限制值，然后选择添加 ulimit。

24. （可选）对于卷配置，请选择添加卷以创建要传递到容器的卷列表。输入卷的名称和源路径，然后选择添加卷。您也可以选择开启启用 EFS。

25. （可选）对于 Tmpfs，请选择添加 tmpfs 以添加 `tmpfs` 挂载。

26. （可选）在日志记录配置部分：

a. 对于日志驱动程序，请选择要使用的日志驱动程序。有关可用日志驱动程序的更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 logDriver。

> **Note**
>
> 默认情况下，使用 `awslogs` 日志驱动程序。

b. 在选项中，选择添加选项以添加选项。输入名称-值对，然后选择添加选项。

c. 对于密钥，选择添加密钥。输入名称-值对，然后选择添加密钥以添加密钥。

> **Tip**
>
> 有关更多信息，请参阅 的 Job 定义参数 ContainerProperties 中的 secretOptions。

27. 选择下一页。
28. 对于作业定义查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建作业定义。

使用创建作业定义 ContainerProperties

以下是包含单个容器的空作业定义模板。您可以使用此模板来创建作业定义，然后将其保存到文件中并与 AWS CLI `--cli-input-json`选项一起使用。有关这些参数的更多信息，请参阅 [的 Job 定义参数 ContainerProperties](#)。

```
{
    "jobDefinitionName": "",
    "type": "container",
    "parameters": {
        "KeyName": ""
    },
    "schedulingPriority": 0,
    "containerProperties": {
        "image": "",
        "vcpus": 0,
        "memory": 0,
        "command": [
            ""
        ],
        "jobRoleArn": "",
        "executionRoleArn": "",
        "volumes": [
            {
                "host": {
                    "sourcePath": ""
                },
                "name": "",
                "efsVolumeConfiguration": {
                    "fileSystemId": "",
                    "rootDirectory": "",
                    "transitEncryption": "ENABLED",
                    "transitEncryptionPort": 0,
                    "authorizationConfig": {
                        "accessPointId": "",
                        "iam": "DISABLED"
                    }
```

```
                }
            }
        ],
        "environment": [
            {
                "name": "",
                "value": ""
            }
        ],
        "mountPoints": [
            {
                "containerPath": "",
                "readOnly": true,
                "sourceVolume": ""
            }
        ],
        "readonlyRootFilesystem": true,
        "privileged": true,
        "ulimits": [
            {
                "hardLimit": 0,
                "name": "",
                "softLimit": 0
            }
        ],
        "user": "",
        "instanceType": "",
        "resourceRequirements": [
            {
                "value": "",
                "type": "MEMORY"
            }
        ],
        "linuxParameters": {
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": [
                        "WRITE"
                    ]
                }
            ],
            "initProcessEnabled": true,
```

```
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [
                        ""
                    ]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "logConfiguration": {
            "logDriver": "syslog",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
                    "name": "",
                    "valueFrom": ""
                }
            ]
        },
        "secrets": [
            {
                "name": "",
                "valueFrom": ""
            }
        ],
        "networkConfiguration": {
            "assignPublicIp": "DISABLED"
        },
        "fargatePlatformConfiguration": {
            "platformVersion": ""
        }
    },
    "nodeProperties": {
        "numNodes": 0,
        "mainNode": 0,
        "nodeRangeProperties": [
            {
                "targetNodes": "",
```

```
                "container": {
                    "image": "",
                    "vcpus": 0,
                    "memory": 0,
                    "command": [
                        ""
                    ],
                    "jobRoleArn": "",
                    "executionRoleArn": "",
                    "volumes": [
                        {
                            "host": {
                                "sourcePath": ""
                            },
                            "name": "",
                            "efsVolumeConfiguration": {
                                "fileSystemId": "",
                                "rootDirectory": "",
                                "transitEncryption": "DISABLED",
                                "transitEncryptionPort": 0,
                                "authorizationConfig": {
                                    "accessPointId": "",
                                    "iam": "ENABLED"
                                }
                            }
                        }
                    ],
                    "environment": [
                        {
                            "name": "",
                            "value": ""
                        }
                    ],
                    "mountPoints": [
                        {
                            "containerPath": "",
                            "readOnly": true,
                            "sourceVolume": ""
                        }
                    ],
                    "readonlyRootFilesystem": true,
                    "privileged": true,
                    "ulimits": [
                        {
```

```
                "hardLimit": 0,
                "name": "",
                "softLimit": 0
            }
        ],
        "user": "",
        "instanceType": "",
        "resourceRequirements": [
            {
                "value": "",
                "type": "MEMORY"
            }
        ],
        "linuxParameters": {
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": [
                        "WRITE"
                    ]
                }
            ],
            "initProcessEnabled": true,
            "sharedMemorySize": 0,
            "tmpfs": [
                {
                    "containerPath": "",
                    "size": 0,
                    "mountOptions": [
                        ""
                    ]
                }
            ],
            "maxSwap": 0,
            "swappiness": 0
        },
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "KeyName": ""
            },
            "secretOptions": [
                {
```

```
                            "name": "",
                            "valueFrom": ""
                        }
                    ]
                },
                "secrets": [
                    {
                        "name": "",
                        "valueFrom": ""
                    }
                ],
                "networkConfiguration": {
                    "assignPublicIp": "DISABLED"
                },
                "fargatePlatformConfiguration": {
                    "platformVersion": ""
                }
            }
        }
    ]
},
"retryStrategy": {
    "attempts": 0,
    "evaluateOnExit": [
        {
            "onStatusReason": "",
            "onReason": "",
            "onExitCode": "",
            "action": "RETRY"
        }
    ]
},
"propagateTags": true,
"timeout": {
    "attemptDurationSeconds": 0
},
"tags": {
    "KeyName": ""
},
"platformCapabilities": [
    "EC2"
],
"eksProperties": {
    "podProperties": {
```

```
            "serviceAccountName": "",
            "hostNetwork": true,
            "dnsPolicy": "",
            "containers": [
                {
                    "name": "",
                    "image": "",
                    "imagePullPolicy": "",
                    "command": [
                        ""
                    ],
                    "args": [
                        ""
                    ],
                    "env": [
                        {
                            "name": "",
                            "value": ""
                        }
                    ],
                    "resources": {
                        "limits": {
                            "KeyName": ""
                        },
                        "requests": {
                            "KeyName": ""
                        }
                    },
                    "volumeMounts": [
                        {
                            "name": "",
                            "mountPath": "",
                            "readOnly": true
                        }
                    ],
                    "securityContext": {
                        "runAsUser": 0,
                        "runAsGroup": 0,
                        "privileged": true,
                        "readOnlyRootFilesystem": true,
                        "runAsNonRoot": true
                    }
                }
            ],
```

```
            "volumes": [
                {
                    "name": "",
                    "hostPath": {
                        "path": ""
                    },
                    "emptyDir": {
                        "medium": "",
                        "sizeLimit": ""
                    },
                    "secret": {
                        "secretName": "",
                        "optional": true
                    }
                }
            ]
        }
    }
}
```

> **Note**
>
> 您可以使用以下命令生成单容器作业定义模板： AWS CLI
>
> ```
> $ aws batch register-job-definition --generate-cli-skeleton
> ```

的 Job 定义参数 ContainerProperties

使用ContainerProperties的 Job 定义分为几个部分：

- 作业定义名称
- 作业定义的类型
- 参数替换占位符默认值
- 作业的容器属性
- 在 Amazon EKS 资源上运行的作业所需的作业定义的 Amazon EKS 属性
- 多节点并行作业所需的节点属性
- 在 Fargate 资源上运行的作业所需的平台功能

- 作业定义的默认标签传播详细信息
- 作业定义的默认重试策略
- 作业定义的默认计划优先级
- 作业定义的默认标签
- 作业定义的默认超时

目录

作业定义名称

`jobDefinitionName`

注册作业定义时，需要指定一个名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。使用该名称注册的第一个作业定义的修订版本为 1。任何使用该名称注册的后续作业定义都会有一个增量修订号。

类型：字符串

必需：是

类型

`type`

当您注册作业定义时，需要指定作业类型。如果作业在 Fargate 资源上运行，则不支持`multinode`。有关多节点并行作业的更多信息，请参阅the section called “创建多节点并行作业定义”。

类型：字符串

有效值：`container` | `multinode`

必需：是

参数

`parameters`

提交作业时，可以指定应替换占位符或覆盖默认作业定义参数的参数。作业提交请求中的参数优先于作业定义中的默认值。这意味着可以对使用相同格式的多个作业使用相同的作业定义。还可以在提交时以编程方式更改命令中的值。

类型：字符串到字符串映射

必需：否

注册作业定义时，可以在作业容器属性的`command`字段中使用参数替代占位符。语法如下所示。

```
"command": [
    "ffmpeg",
    "-i",
    "Ref::inputfile",
    "-c",
    "Ref::codec",
    "-o",
    "Ref::outputfile"
]
```

在上面的示例中，命令中包含参数替代占位符*`Ref::inputfile`*、*`Ref::codec`*和*`Ref::outputfile`*。您可以使用作业定义中

的parameters对象为这些占位符设置默认值。例如，要为*Ref::codec*占位符设置默认值，可以在作业定义中指定以下内容：

```
"parameters" : {"codec" : "mp4"}
```

在提交此作业定义以运行时，容器命令中的*Ref::codec*参数将被替换为默认值。mp4

容器属性

注册作业定义时，指定容器属性列表，在置放作业时，需要将这些容器属性传递给容器实例上的 Docker 进程守护程序。作业定义中允许使用以下容器属性。对于单节点作业，这些容器属性是在作业定义级别设置的。对于多节点并行作业，每个节点组的容器属性是在节点属性级别设置的。

command

传递给容器的命令。此参数映射到 Docker Remote API 创建容器部分中的Cmd，以及docker run的COMMAND参数。有关 Docker CMD参数的更多信息，请参阅 https://docs.docker.com/engine/reference/builder/#cmd。

```
"command": ["string", ...]
```

类型：字符串数组

必需：否

environment

要传递给容器的环境变量。此参数将映射到 Docker Remote API 的创建容器部分中的Env以及docker run的--env选项。

> **Important**
>
> 建议不要对敏感信息（如凭证数据）使用纯文本环境变量。

> **Note**
>
> 环境变量不得以AWS_BATCH开头。此命名约定是为 AWS Batch 服务设置的变量保留的。

类型：键/值对的数组

必需：否

name

环境变量的名称。

类型：字符串

必需：是，当使用environment时。

value

环境变量的值。

类型：字符串

必需：是，当使用environment时。

```
"environment" : [
    { "name" : "envName1", "value" : "envValue1" },
    { "name" : "envName2", "value" : "envValue2" }
]
```

executionRoleArn

注册作业定义时，可以指定 IAM 角色。该角色为 Amazon ECS 容器代理提供了代表您调用其关联策略中指定的 API 操作的权限。对于在 Fargate 资源上运行的作业，必须提供执行角色。有关更多信息，请参阅 AWS Batch 执行 IAM 角色。

类型：字符串

必需：否

fargatePlatformConfiguration

适用于在 Fargate 资源上运行的作业的平台配置。在 EC2 资源上运行的作业不得指定此参数。

类型：FargatePlatform配置对象

必需：否

platformVersion

AWS Fargate 平台版本用于工作，或者LATEST使用最新批准的 Fargate AWS 平台版本。

类型：字符串

默认：LATEST

必需：否

`image`

用于启动作业的映像。此字符串将直接传递给 Docker 进程守护程序。默认情况下，Docker Hub 注册表中的映像可用。也可以使用*repository-url*/*image*:*tag*指定其他存储库。允许最多 255 个字母（大写和小写字母）、数字、连字符、下划线、冒号、句点、正斜杠和井号。此参数可映射到 Docker Remote API 的创建容器部分中的Image和docker run的IMAGE参数。

> **Note**
>
> Docker 映像架构必须与为它们安排的计算资源的处理器架构相匹配。例如，基于 Arm 的 Docker 映像只能在基于Arm的计算资源上运行。

- Amazon ECR 公有存储库中的映像使用完整的`registry/repository[:tag]`或`registry/repository[@digest]`命名惯例（例如，`public.ecr.aws/`*registry_alias*`/`*my-web-app*`:`*latest*）。
- Amazon ECR 存储库中的映像使用完整的`registry/repository:[tag]`命名惯例。例如，*aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-web-app*`:`*latest*。
- Docker Hub 上的官方存储库中的映像使用一个名称 (例如，ubuntu或mongo)。
- Docker Hub 上其他存储库中的映像通过组织名称 (例如，amazon/amazon-ecs-agent) 进行限定。
- 其他在线存储库中的映像由域名 (例如，quay.io/assemblyline/ubuntu) 进行进一步限定。

类型：字符串

必需：是

`instanceType`

要用于多节点并行作业的实例类型。多节点并行作业中的所有节点组必须使用相同的实例类型。此参数不适用于单节点容器作业或在 Fargate 资源上运行的作业。

类型：字符串

必需：否

`jobRoleArn`

注册作业定义时，可以指定 IAM 角色。该角色为作业容器提供了代表您调用其关联的策略中指定的 API 操作的权限。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的任务的 IAM 角色。

类型：字符串

必需：否

`linuxParameters`

特定于 Linux 的修改应用于容器的详细信息，如设备映射的详细信息。

```
"linuxParameters": {
    "devices": [
        {
            "hostPath": "string",
            "containerPath": "string",
            "permissions": [
                "READ", "WRITE", "MKNOD"
            ]
        }
    ],
    "initProcessEnabled": true|false,
    "sharedMemorySize": 0,
    "tmpfs": [
        {
            "containerPath": "string",
            "size": integer,
            "mountOptions": [
                "string"
            ]
        }
    ],
    "maxSwap": integer,
    "swappiness": integer
}
```

类型：LinuxParameters 对象

必需：否

devices

映射到容器的设备列表。此参数将映射到 Docker Remote API 的创建容器部分中的Devices以及 Docker 运行的--device选项。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：设备对象数组

必需：否

hostPath

主机容器实例中可用设备的路径。

类型：字符串

必需：是

containerPath

在容器中公开设备的路径。如果未指定，则设备将在与主机路径相同的路径上公开。

类型：字符串

必需：否

permissions

容器中设备的权限。如果未指定，则将权限设置为READ、WRITE和MKNOD。

类型：字符串数组

必需：否

有效值：READ |WRITE |MKNOD

initProcessEnabled

如果为 true，则在容器内运行init进程，以转发信号和获得进程。此参数会将--init选项映射到 Docker 运行。此参数要求容器实例上的 Docker Remote API 版本为 1.25 或更高版本。要检查容器实例上的 Docker Remote API 版本，请登录到容器实例并运行以下命令：sudo docker version | grep "Server API version"

类型：布尔值

必需：否

maxSwap

作业可以使用的交换内存总量（以 MiB 为单位）。该参数会转换为 Docker 运行的--memory-swap选项，其中，该值为容器内存和maxSwap值之和。有关更多信息，请参阅 Docker 文档中的--memory-swap详细信息。

如果指定maxSwap值为0，则该容器不使用交换。接受的值为0或任何正整数。如果省略maxSwap参数，该容器将为其运行所在的容器实例使用交换配置。必须为要使用的swappiness参数设置maxSwap值。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：整数

必需：否

sharedMemorySize

/dev/shm卷的大小值 (以 MiB 为单位)。此参数会将--shm-size选项映射到 Docker 运行。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：整数

必需：否

swappiness

可以使用此功能调整容器的内存交换行为。除非绝对必要，否则0的一个swappiness值将导致交换不会发生。swappiness值为100将导致页面被积极地交换。接受的值为0到100之间的整数。如果未指定swappiness参数，则使用默认值60。如果未指定maxSwap的值，则此参数将

被忽略。如果maxSwap设置为 0，则容器不使用交换。此参数会将--memory-swappiness选项映射到 Docker 运行。

在使用每个容器交换配置时，请考虑以下事项。

- 必须在容器实例上启用并分配交换空间才能供容器使用。

> **Note**
>
> 默认情况下，Amazon ECS 优化 AMI 没有启用交换功能。必须在实例上启用交换才能使用此功能。有关更多信息，请参阅 Amazon EC2 用户指南中的实例存储交换卷或如何使用交换文件在 Amazon EC2 实例中分配内存以用作交换空间？ 。

- 只有使用 EC2 资源的作业定义才支持交换空间参数。
- 如果作业定义中忽略了maxSwap和swappiness参数，每个容器都有默认的swappiness值：60。总的交换使用量限制为容器的内存预留量的两倍。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：整数

必需：否

`tmpfs`

tmpfs挂载的容器路径、挂载选项和大小。

类型：Tmpfs 对象数组

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

必需：否

`containerPath`

挂载tmpfs卷的容器中的绝对文件路径。

类型：字符串

必需：是

`mountOptions`

tmpfs卷挂载选项列表。

有效值："defaults" | "ro" | "rw" | "suid" | "nosuid" | "dev" | "nodev" | "exec" | "noexec" | "sync" | "async" | "dirsync" | "remount" | "mand" | "nomand" | "atime" | "noatime" | "diratime" | "nodiratime" | "bind" | "rbind" | "unbindable" | "runbindable" | "private" | "rprivate" | "shared" | "rshared" | "slave" | "rslave" | "relatime" | "norelatime" | "strictatime" | "nostrictatime" | "mode" | "uid" | "gid" | "nr_inodes" | "nr_blocks" | "mpol"

类型：字符串数组

必需：否

`size`

tmpfs卷的大小 (以 MiB 为单位)。

类型：整数

必需：是

`logConfiguration`

作业的日志配置规范。

此参数将映射到 Docker Remote API 的创建容器部分中的`LogConfig`以及 Docker 运行的`--log-driver`选项。默认情况下，容器使用与 Docker 进程守护程序相同的日志记录驱动程序。但容器可能通过在容器定义中使用此参数指定日志驱动程序，以此来使用不同于 Docker 进程守护程序的日志记录驱动程序。要对容器使用另一个日志记录驱动程序，必须在容器实例或另一个日志服务器上配置日志系统，以提供远程日志记录选项。有关其他支持的日志驱动程序选项的更多信息，请参阅 Docker 文档中的配置日志记录驱动程序。

> **Note**
>
> AWS Batch 目前支持 Docker 守护程序可用的日志驱动程序子集（如LogConfiguration数据类型所示）。

此参数要求容器实例上的 Docker Remote API 版本为 1.18 或更高版本。要检查容器实例上的 Docker Remote API 版本，请登录到容器实例并运行以下命令：`sudo docker version | grep "Server API version"`

```
"logConfiguration": {
    "devices": [
        {
            "logDriver": "string",
            "options": {
                "optionName1" : "optionValue1",
                "optionName2" : "optionValue2"
            }
            "secretOptions": [
              {
                  "name" : "secretOptionName1",
                  "valueFrom" : "secretOptionArn1"
              },
              {
                  "name" : "secretOptionName2",
                  "valueFrom" : "secretOptionArn2"
              }
            ]
        }
    ]
}
```

类型：LogConfiguration 对象

必需：否

`logDriver`

要用于作业的日志驱动程序。默认情况下， AWS Batch 启用`awslogs`日志驱动程序。为此参数列出的有效值是默认情况下 Amazon ECS 容器代理可与之通信的日志驱动程序。

此参数将映射到 Docker Remote API 的创建容器部分中的`LogConfig`以及 Docker 运行的`--log-driver`选项。默认情况下，作业使用与 Docker 进程守护程序相同的日志记录驱动程序。但作业可能通过在作业定义中使用此参数指定日志驱动程序，以此来使用不同于 Docker 进程守护程序的日志记录驱动程序。如果要为作业指定另一个日志记录驱动程序，则必须在计算环境中的容器实例上配置日志系统。或者，也可以在另一台日志服务器上对其进行配置，以提供远程日志记录选项。有关其他支持的日志驱动程序选项的更多信息，请参阅 Docker 文档中的配置日志记录驱动程序。

> **Note**
>
> AWS Batch 目前支持 Docker 守护程序可用的日志驱动程序子集。可能会在 Amazon ECS 容器代理的未来版本中提供其他日志驱动程序。

支持的日志驱动程序为`awslogs`、`fluentd`、`gelf`、`json-file`、`journald`、`logentries`、`syslog`和`splunk`。

> **Note**
>
> 在 Fargate 资源上运行的作业仅限于`awslogs`和`splunk`日志驱动程序。

此参数要求容器实例上的 Docker Remote API 版本为 1.18 或更高版本。要检查容器实例上的 Docker Remote API 版本，请登录到容器实例并运行以下命令：`sudo docker version | grep "Server API version"`

> **Note**
>
> 在容器实例上运行的 Amazon ECS 容器代理必须将该实例上的可用日志记录驱动程序注册到`ECS_AVAILABLE_LOGGING_DRIVERS`环境变量。否则，放置在该实例上的容器将无法使用这些日志配置选项。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 容器代理配置。

`awslogs`

指定 Amazon CloudWatch 日志记录驱动程序。有关更多信息，请参阅 Docker 文档中的使用 awslogs 日志驱动程序和 Amazon CloudWatch Logs 日志驱动程序。

`fluentd`

指定 Fluentd 日志记录驱动程序。有关更多信息（包括用法和选项），请参阅 Docker 文档中的 Fluentd 日志记录驱动程序。

`gelf`

指定 Graylog Extended Format (GELF) 日志记录驱动程序。有关更多信息（包括用法和选项），请参阅 Docker 文档中的 Graylog Extended Format 日志记录驱动程序。

`journald`

指定 journald 日志记录驱动程序。有关更多信息（包括用法和选项），请参阅 Docker 文档中的 Journald 日志记录驱动程序。

`json-file`

指定 JSON 文件日志记录驱动程序。有关更多信息（包括用法和选项），请参阅 Docker 文档中的 JSON 文件日志记录驱动程序。

`splunk`

指定 Splunk 日志记录驱动程序。有关更多信息（包括用法和选项），请参阅 Docker 文档中的 Splunk 日志记录驱动程序。

`syslog`

指定 syslog 日志记录驱动程序。有关更多信息（包括用法和选项），请参阅 Docker 文档中的 Syslog 日志记录驱动程序。

类型：字符串

必需：是

有效值：`awslogs` | `fluentd` | `gelf` | `journald` | `json-file` | `splunk` | `syslog`

> **Note**
>
> 如果您想要与 Amazon ECS 容器代理一起使用之前未列出的自定义驱动程序，则可以分叉可用的 Amazon ECS 容器代理项目，GitHub然后对其进行自定义以使用该驱动程序。我们鼓励您针对要包含的更改提交拉取请求。但是，Amazon Web Services 当前不支持运行此软件修改后副本的请求。

`options`

要发送到作业的日志驱动程序中的日志配置选项。

此参数要求容器实例上的 Docker Remote API 版本为 1.19 或更高版本。

类型：字符串到字符串映射

必需：否

`secretOptions`

表示要传递到日志配置的密文的对象。有关更多信息，请参阅 指定敏感数据。

类型：对象数组

必需：否

`name`

要在作业中设置的日志驱动程序选项的名称。

类型：字符串

必需：是

`valueFrom`

要向容器的日志配置公开的秘钥的 Amazon 资源名称（ARN）。支持的值为 Secrets Manager 秘钥的完整 ARN 或 SSM Parameter Store 中参数的完整 ARN。

> **Note**
>
> 如果 SSM Parameter Store 参数与您正在启动的任务相同 AWS 区域 ，则可以使用该参数的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

类型：字符串

必需：是

`memory`

已弃用此参数，请改用*resourceRequirements*。

为作业预留的 MiB 内存的数量。

例如，如果您的作业定义包含类似于以下内容的语法，则说明如何使用resourceRequirements。

```
"containerProperties": {
```

```
  "memory": 512
}
```

使用resourceRequirements的等效语法如下。

```
"containerProperties": {
  "resourceRequirements": [
    {
      "type": "MEMORY",
      "value": "512"
    }
  ]
}
```

类型：整数

必需：是

mountPoints

容器中数据卷的挂载点。此参数将映射到 Docker Remote API 的创建容器部分中的 Volumes 以及 docker run 的 --volume 选项。

```
"mountPoints": [
                {
                  "sourceVolume": "string",
                  "containerPath": "string",
                  "readOnly": true|false
                }
              ]
```

类型：对象数组

必需：否

sourceVolume

要挂载的卷的名称。

类型：字符串

必需：是，当使用mountPoints时。

containerPath

要将主机卷挂载到的容器上的路径。

类型：字符串

必需：是，当使用mountPoints时。

readOnly

如果此值为true，则容器具有对卷的只读访问权。如果此值为false，则容器可对卷进行写入。

类型：布尔值

必需：否

默认值：False

networkConfiguration

适用于在 Fargate 资源上运行的作业的网络配置。在 EC2 资源上运行的作业不得指定此参数。

```
"networkConfiguration": {
   "assignPublicIp": "string"
}
```

类型：对象数组

必需：否

assignPublicIp

指示作业是否具有公有 IP 地址。如果作业需要出站网络访问权限，则必须这样做。

类型：字符串

有效值：ENABLED | DISABLED

必需：否

默认：DISABLED

`privileged`

当此参数为 true 时，将对此容器提供对主机容器实例的提升的权限（类似于`root`用户）。此参数将映射到 Docker Remote API 的创建容器部分中的`Privileged`以及docker run的`--privileged`选项。此参数不适用于在 Fargate 资源上运行的作业。请勿提供或将其指定为 false。

```
"privileged": true|false
```

类型：布尔值

必需：否

`readonlyRootFilesystem`

当此参数为 true 时，将对此容器提供对其根文件系统的只读访问权。此参数将映射到 Docker Remote API 的创建容器部分中的 `ReadonlyRootfs` 以及 docker run 的 `--read-only` 选项。

```
"readonlyRootFilesystem": true|false
```

类型：布尔值

必需：否

`resourceRequirements`

要分配给容器的资源的类型和数量。支持的资源包括`GPU`、`MEMORY`和`VCPU`。

```
"resourceRequirements" : [
  {
    "type": "GPU",
    "value": "number"
  }
]
```

类型：对象数组

必需：否

`type`

要分配给容器的资源类型。支持的资源包括`GPU`、`MEMORY`和`VCPU`。

类型：字符串

必需：是，当使用`resourceRequirements`时。

`value`

要为容器预留的指定资源的数量。这些值根据`type`指定的不同而有所不同。

type="GPU"

为容器预留的物理 GPU 数量。为作业中所有容器预留的 GPU 数量不能超过启动作业的计算资源上可用 GPU 的数量。

type="MEMORY"

要提供给容器的内存的硬限制（以 MiB 为单位）。如果容器尝试使用超出此处指定的内存，该容器将被终止。此参数将映射到 Docker Remote API 的 Create a container（创建容器）部分中的 `Memory` 以及 docker run 的 `--memory` 选项。您必须为作业指定至少 4 MiB 内存。对于多节点并行 (MNP) 作业来说，这是必需的，但可以在多个位置指定。必须至少为每个节点指定一次。此参数将映射到 Docker Remote API 的 Create a container（创建容器）部分中的 `Memory` 以及 docker run 的 `--memory` 选项。

> **Note**
>
> 如果要通过为特定实例类型的作业提供尽可能多的内存来最大化资源利用率，请参阅计算资源内存管理。

对于在 Fargate 资源上运行的作业，`value`必须与受支持的值之一匹配。此外，这些VCPU值必须是该内存值支持的值之一。

VCPU	MEMORY
0.25 个 vCPU	512、1024 和 2048 MiB
0.5 个 vCPU	1024-4096 MiB 以 1024 MiB 为增量
1 个 vCPU	2048-8192 MiB 以 1024 MiB 为增量
2 个 vCPU	4096-16384 MiB 以 1024 MiB 为增量
4 个 vCPU	8192-30720 MiB 以 1024 MiB 为增量

VCPU	MEMORY
8 个 vCPU	16384-61440 MiB 以 4096 MiB 为增量
16 个 vCPU	32768-122880 MiB 以 8192 MiB 为增量

type="VCPU"

为作业预留的 vCPU 的数量。此参数将映射到 Docker Remote API 的 Create a container（创建容器）部分中的 CpuShares 以及 docker run 的 --cpu-shares 选项。每个 vCPU 相当于 1024 个 CPU 份额。对于在 EC2 资源上运行的作业，必须至少指定一个 vCPU。这是必需的，但可以在多个位置指定。必须至少为每个节点指定一次。

对于在 Fargate 资源上运行的作业，value必须与受支持的值之一匹配，且MEMORY值必须是该 VCPU 值支持的值之一。支持的值为 0.25、0.5、1、2、4、8 和 16。

Fargate 按需 vCPU 资源计数配额的默认值为 6 个 vCPU。有关 Fargate 配额的更多信息，请参阅Amazon Web Services 一般参考中AWS Fargate 配额。

类型：字符串

必需：是，当使用resourceRequirements时。

secrets

作为环境变量公开的作业密文。有关更多信息，请参阅 指定敏感数据。

```
"secrets": [
    {
      "name": "secretName1",
      "valueFrom": "secretArn1"
    },
    {
      "name": "secretName2",
      "valueFrom": "secretArn2"
    }
    ...
]
```

类型：对象数组

必需：否

`name`

包含密文的环境变量的名称。

类型：字符串

必需：是，当使用`secrets`时。

`valueFrom`

要向容器公开的密文。支持的值是 Secrets Manager 密文的完整 Amazon 资源名称（ARN）或 SSM Parameter Store 中参数的完整 ARN。

> **Note**
>
> 如果 SSM Parameter Store 参数与您启动的任务 AWS 区域 相同，则可以使用该参数的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

类型：字符串

必需：是，当使用`secrets`时。

`ulimits`

要在容器中设置的`ulimits`值的列表。此参数将映射到 Docker Remote API 的创建容器部分中的 `Ulimits` 以及 docker run 的 `--ulimit` 选项。

```
"ulimits": [
  {
    "name": string,
    "softLimit": integer,
    "hardLimit": integer
  }
  ...
]
```

类型：对象数组

必需：否

name

ulimit的type。

类型：字符串

必需：是，当使用ulimits时。

hardLimit

ulimit类型的硬限制。

类型：整数

必需：是，当使用ulimits时。

softLimit

ulimit类型的软限制。

类型：整数

必需：是，当使用ulimits时。

user

要在容器内使用的用户名。此参数将映射到 Docker Remote API 的创建容器部分中的 User 以及 docker run 的 --user 选项。

```
"user": "string"
```

类型：字符串

必需：否

vcpus

已弃用此参数，请改用*resourceRequirements*。

为容器预留的 vCPU 的数量。

作为resourceRequirements使用方法的示例，如果作业定义包含类似于以下的行：

```
"containerProperties": {
  "vcpus": 2
}
```

使用[resourceRequirements]的等效行如下所示。

```
"containerProperties": {
  "resourceRequirements": [
    {
      "type": "VCPU",
      "value": "2"
    }
  ]
}
```

类型：整数

必需：是

volumes

注册作业定义时，可以指定需传递给容器实例上的 Docker 进程守护程序的卷的列表。容器属性中允许以下参数：

```
"volumes": [
  {
    "name": "string",
    "host": {
      "sourcePath": "string"
    },
    "efsVolumeConfiguration": {
      "authorizationConfig": {
        "accessPointId": "string",
        "iam": "string"
      },
      "fileSystemId": "string",
      "rootDirectory": "string",
      "transitEncryption": "string",
      "transitEncryptionPort": number
    }
  }
]
```

name

卷的名称。最多能包含 255 个字母 (大写和小写字母)、数字、连字符和下划线。此名称已在容器定义sourceVolume的mountPoints参数中引用。

类型：字符串

必需：否

host

host参数的内容确定数据卷是否一直保存在主机容器实例上以及存储它的位置上。如果host参数为空，则 Docker 进程守护程序将为数据卷分配一个主机路径。但是，在与该卷关联的容器停止运行后，不保证保存数据。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：对象

必需：否

sourcePath

向容器提供的主机容器实例上的路径。如果此参数为空，则 Docker 进程守护程序将分配一个主机路径。

如果host参数包含sourcePath文件位置，则数据卷将在主机容器实例上的指定位置保留，除非手动将其删除。如果主机容器实例上不存在sourcePath值，则 Docker 进程守护程序将创建该值。如果该位置不存在，则将导出源路径文件夹的内容。

类型：字符串

必需：否

efsVolumeConfiguration

使用 Amazon Elastic File System 文件系统进行任务存储时，指定此参数。有关更多信息，请参阅 Amazon EFS 卷。

类型：对象

必需：否

`authorizationConfig`

Amazon EFS 文件系统的授权配置详细信息。

类型：字符串

必需：否

`accessPointId`

要使用的 Amazon EFS 接入点 ID。如果指定了接入点，则必须省略在`EFSVolumeConfiguration`中指定的根目录值，或者将其设置为`/`。这将强制执行 EFS 接入点上设置的路径。如果使用接入点，则必须在`EFSVolumeConfiguration`中启用传输加密。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的使用 Amazon EFS 接入点。

类型：字符串

必需：否

`iam`

确定在装载 Amazon EFS 文件系统时是否使用任务定义中定义的任务 IAM 角色。 AWS Batch 如果启用，则必须在`EFSVolumeConfiguration`中启用传输加密。如果忽略此参数，将使用默认值`DISABLED`。有关更多信息，请参阅 使用 Amazon EFS 接入点。

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

`fileSystemId`

要使用的 Amazon EFS 文件系统 ID。

类型：字符串

必需：否

`rootDirectory`

Amazon EFS 文件系统中要作为主机内的根目录挂载的目录。如果忽略此参数，将使用 Amazon EFS 卷的根目录。如果指定，`/`这与忽略此参数效果相同。最大长度为 4096 个字符。

> **Important**
>
> 如果在authorizationConfig中指定了 EFS 接入点，则必须省略根目录参数，或者将其设置为/。这将在 Amazon EFS 接入点上强制执行设置的路径。

类型：字符串

必需：否

`transitEncryption`

确定是否对 Amazon ECS 主机和 Amazon EFS 服务器之间传输的 Amazon EFS 数据启用加密。如果使用 Amazon EFS IAM 授权，则必须启用传输加密。如果忽略此参数，将使用默认值DISABLED。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的加密传输中数据。

类型：字符串

有效值：ENABLED | DISABLED

必需：否

`transitEncryptionPort`

在 Amazon ECS 主机和 Amazon EFS 服务器之间发送加密数据时要使用的端口。如果未指定传输加密端口，将使用 Amazon EFS 挂载帮助程序使用的端口选择策略。该值必须在 0 到 65535 之间。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 EFS 挂载帮助程序。

类型：整数

必需：否

Amazon EKS 属性

具有各种属性的对象，这些属性特定于基于 Amazon EKS 的作业。不得为基于 Amazon ECS 的作业定义指定此项。

`podProperties`

作业的Kubernetes容器组（pod）资源的属性。

类型：EksPod属性对象

必需：否

`containers`

在 Amazon EKS 容器组上使用的容器属性。

类型：EksContainer 对象

必需：否

`args`

入口点的参数数组。如果未指定，则使用容器映像的CMD。这对应Kubernetes中容器组（pod）入口点 部分的args成员。使用容器的环境扩展环境变量引用。

如果引用的环境变量不存在，则命令中的引用不会更改。例如，如果引用是“$(NAME1)”且NAME1环境变量不存在，则命令字符串将保留“$(NAME1)”。$$替换为$，并且生成的字符串未扩展。例如，无论VAR_NAME环境变量是否存在，$$(VAR_NAME)都会作为$(VAR_NAME)传递。有关更多信息，请参阅《Dockerfile 参考》中的 CMD 和Kubernetes文档中的为容器组（pod）定义命令和参数。

类型：字符串数组

必需：否

`command`

容器的入口点。这不是在 Shell 中运行。如果未指定，则使用容器映像的ENTRYPOINT。使用容器的环境扩展环境变量引用。

如果引用的环境变量不存在，则命令中的引用不会更改。例如，如果引用是“$(NAME1)”且NAME1环境变量不存在，则命令字符串将保留“$(NAME1)”。$$替换为$，并且生成的字符串未扩展。例如，无论VAR_NAME环境变量是否存在，$$(VAR_NAME)都会作为$(VAR_NAME)传递。无法更新入口点。有关更多信息，请参阅 《Dockerfile 参考》中的入口点和 Kubernetes文档中的为容器设置启动时要执行的命令和参数和入口点。

类型：字符串数组

必需：否

`env`

要传递给容器的环境变量。

> **Note**
>
> 环境变量不得以 AWS_BATCH 开头。此命名约定是为 AWS Batch 设置的变量保留的。

类型：EksContainerEnvironmentVariable 对象数组

必需：否

name

环境变量的名称。

类型：字符串

必需：是

value

环境变量的值。

类型：字符串

必需：否

image

用于启动容器的 Docker 映像。

类型：字符串

必需：是

imagePullPolicy

容器的图像提取策略。支持的值有Always、IfNotPresent和Never。此参数默认为IfNotPresent。但是，如果指定了:latest标签，则默认为Always。有关更多信息，请参阅Kubernetes文档中的更新映像。

类型：字符串

必需：否

name

容器的名称。如果未指定名称，则使用默认名称“Default”。容器组中的每个容器必须具有唯一的名称。

类型：字符串

必需：否

resources

要分配给容器的资源的类型和数量。支持的资源包括memory、cpu和nvidia.com/gpu。有关更多信息，请参阅Kubernetes文档中的容器组（pod）和容器的资源管理。

类型：EksContainerResourceRequirements 对象

必需：否

limits

为容器预留的资源类型和数量。这些值因指定的name而异。可以使用limits或requests对象请求资源。

memory

容器的内存硬限值（以 MiB 为单位）用整数表示，带“Mi”后缀。如果容器试图超出指定的内存，则容器将终止。必须为作业指定至少 4MiB 的内存。可以在limits、requests或两者中指定memory。如果在这两个位置中指定了memory，则limits中指定的值必须等于requests中指定的值。

> **Note**
>
> 为了最大程度地利用资源，请为作业提供尽可能多的内存，以用于正在使用的具体实例类型。要了解如何操作，请参阅 计算资源内存管理。

cpu

为容器预留的 CPU 的数量。值必须是0.25的偶数倍。可以在limits、requests或两者中指定cpu。如果在这两个位置中指定了cpu，则limits中指定的值必须至少与requests中指定的值一样大。

nvidia.com/gpu

为容器预留的 GPU 的数量。值必须是整数。可以在limits、requests或两者中指定memory。如果在这两个位置中指定了 memory，则 limits 中指定的值必须等于 requests 中指定的值。

类型：字符串到字符串映射

值长度限制：最小长度为 1。最大长度为 256。

必需：否

requests

为容器请求的资源类型和数量。这些值因指定的name而异。可以使用limits或requests对象请求资源。

memory

容器的内存硬限值（以 MiB 为单位）用整数表示，带“Mi”后缀。如果容器试图超出指定的内存，则容器将终止。必须为作业指定至少 4MiB 的内存。可以在limits、requests或两者中指定memory。如果在两者中均指定了memory，则limits中指定的值必须等于requests中指定的值。

> **Note**
>
> 如果要通过为特定实例类型的作业提供尽可能多的内存来最大化资源利用率，请参阅计算资源内存管理。

cpu

为容器预留的 CPU 的数量。值必须是0.25的偶数倍。可以在limits、requests或两者中指定cpu。如果在两者中均指定了cpu，则limits中指定的值必须至少与requests中指定的值一样大。

nvidia.com/gpu

为容器预留的 GPU 的数量。值必须是整数。可以在limits、requests或两者中指定nvidia.com/gpu。如果在两者中均指定了 nvidia.com/gpu，则 limits 中指定的值必须等于 requests 中指定的值。

类型：字符串到字符串映射

值长度限制：最小长度为 1。最大长度为 256。

必需：否

`securityContext`

作业的安全上下文。有关更多信息，请参阅Kubernetes文档中的为容器组（pod）或容器配置安全上下文。

类型：EksContainerSecurityContext 对象

必需：否

`privileged`

当此参数为`true`时，将对此容器提供对主机容器实例的提升权限。权限级别与`root`用户权限类似。默认值为`false`。该参数映射到Kubernetes文档中特权容器组（pod）安全策略中的`privileged`策略。

类型：布尔值

必需：否

`readOnlyRootFilesystem`

当此参数为`true`时，将对此容器提供对其根文件系统的只读访问权。默认值为`false`。该参数映射到Kubernetes文档中卷和文件系统容器组（pod）安全策略中的`ReadOnlyRootFilesystem`策略。

类型：布尔值

必需：否

`runAsGroup`

指定此参数时，容器将作为指定的组 ID (`gid`) 运行。如果未指定此参数，则默认为映像元数据中指定的群组。该参数映射到Kubernetes文档中用户和群组容器组（pod）安全策略中的`RunAsGroup`和`MustRunAs`策略。

类型：长整型

必需：否

runAsNonRoot

指定此参数时，容器是以uid用户（非 0）身份运行的。如果未指定此参数，则会强制执行此规则。该参数映射到Kubernetes文档中用户和群组容器组（pod）安全策略中的RunAsUser和MustRunAsNonRoot策略。

类型：长整型

必需：否

runAsUser

指定此参数时，容器将作为指定的用户 ID (uid) 运行。如果未指定此参数，则默认为映像元数据中指定的用户。该参数映射到Kubernetes文档中用户和群组容器组（pod）安全策略中的RunAsUser和MustRanAs策略。

类型：长整型

必需：否

volumeMounts

用于 Amazon EKS 作业容器的卷挂载。有关Kubernetes中卷和卷挂载的更多信息，请参阅Kubernetes文档中的卷。

类型：EksContainerVolumeMount 对象数组

必需：否

mountPath

挂载卷的容器上的路径。

类型：字符串

必需：否

name

卷挂载的名称。这必须与容器组中其中一个卷的名称相匹配。

类型：字符串

必需：否

`readOnly`

如果此值为`true`，则容器具有对卷的只读访问权。否则，容器可以写入卷。默认值为`false`。

类型：布尔值

必需：否

`dnsPolicy`

容器组的 DNS 策略。默认值为 `ClusterFirst`。如果未指定`hostNetwork`参数，则默认值为`ClusterFirstWithHostNet`。`ClusterFirst`指示任何与配置的集群域后缀不匹配的 DNS 查询都将转发到继承自节点的上游名称服务器。如果在RegisterJob定义 API 操作`dnsPolicy`中未指定任何值，则DescribeJob定义或 DescribeJobsAPI 操作都不会返回任何值。`dnsPolicy`容器组规格设置将包含`ClusterFirst`或`ClusterFirstWithHostNet`，具体取决于`hostNetwork`参数的值。有关更多信息，请参阅Kubernetes文档中的容器组（pod）的 DNS 策略。

有效值：`Default` |`ClusterFirst` |`ClusterFirstWithHostNet`

类型：字符串

必需：否

`hostNetwork`

指示容器组是否使用主机的网络 IP 地址。默认值为 `true`。将其设置为`false`将启用 Kubernetes容器组（pod）联网模型。大多数 AWS Batch 工作负载仅限出口，不需要为每个 pod 分配传入连接的 IP 开销。有关更多信息，请参阅Kubernetes文档中的主机命名空间和容器组（pod）网络。

类型：布尔值

必需：否

`serviceAccountName`

用于运行容器组的服务账户的名称。有关更多信息，请参阅《Amazon EKS 用户指南》中的Kubernetes服务账户和配置Kubernetes服务账户以代入 IAM 角色，以及Kubernetes文档中的为容器组（pod）配置服务账号。

类型：字符串

必需：否

volumes

为使用 Amazon EKS 资源的作业定义指定卷。

类型：EksVolume 对象数组

必需：否

emptyDir

指定Kubernetes emptyDir卷的配置。将容器组分配给节点时，会先创建emptyDir卷。只要容器组（pod）在该节点上运行，该卷就会存在。emptyDir 卷最初是空的。容器组中的所有容器都可以读取和写入emptyDir卷中的文件。但是，emptyDir卷可以挂载在每个容器中的相同或不同的路径上。出于任何原因将容器组从节点中移除时，将永久删除emptyDir中的数据。有关更多信息，请参阅Kubernetes文档中的 emptyDir。

类型：EksEmpty目录对象

必需：否

medium

存储卷的介质。默认值为空字符串，该字符串使用节点的存储。

""

（默认）使用节点的磁盘存储。

“Memory”

使用由节点 RAM 支持的tmpfs卷。节点重新启动时，卷的内容会丢失，并且该卷上的任何存储都将计入容器的内存限制。

类型：字符串

必需：否

sizeLimit

卷的最大大小。默认情况下，未定义最大大小。

类型：字符串

长度限制：最小长度为 1。最大长度为 256。

必需：否

hostPath

指定Kubernetes `hostPath`卷的配置。`hostPath`卷将主机节点文件系统中的现有文件或目录挂载到容器组中。有关更多信息，请参阅Kubernetes文档中的 hostPath。

类型：EksHost路径对象

必需：否

path

主机上要挂载到容器组中的文件或目录的路径。

类型：字符串

必需：否

name

卷的名称。必须允许该名称作为 DNS 子域名。有关更多信息，请参阅Kubernetes文档中的 DNS 子域名。

类型：字符串

必需：是

secret

指定Kubernetes `secret`卷的配置。有关更多信息，请参阅Kubernetes文档中的密文。

类型：EksSecret 对象

必需：否

可选

指定是否必须定义密文或密文的密钥。

类型：布尔值

必需：否

secretName

密文的名称。必须允许该名称作为 DNS 子域名。有关更多信息，请参阅Kubernetes文档中的 DNS 子域名。

类型：字符串

必需：是

平台功能

`platformCapabilities`

作业定义所需的平台功能。如果未指定任何值，则默认为`EC2`。对于在 Fargate 资源上运行的作业，指定了`FARGATE`。

> **Note**
>
> 如果作业在 Amazon EKS 资源上运行，则不得指定`platformCapabilities`。

类型：字符串

有效值：`EC2` | `FARGATE`

必需：否

传播标签

`propagateTags`

指定是否将标签从作业或作业定义传播到相应的 Amazon ECS 任务。如果未指定任何值，则不会传播标签。只能在创建任务后将标签传播到任务。对于名称相同的标签，作业标签的优先级高于作业定义标签。如果作业和作业定义的组合标签总数超过 50 个，则作业将转为`FAILED`状态。

> **Note**
>
> 如果作业在 Amazon EKS 资源上运行，则不得指定`propagateTags`。

类型：布尔值

必需：否

节点属性

`nodeProperties`

注册多节点并行作业定义时，必须指定节点属性的列表。这些节点属性定义了作业中要使用的节点数量、主节点索引以及要使用的不同节点范围。如果作业在 Fargate 资源上运行，则不能指定`nodeProperties`。请改用`containerProperties`。作业定义中允许使用以下节点属性。有关更多信息，请参阅 多节点并行作业。

> **Note**
>
> 如果作业在 Amazon EKS 资源上运行，则不得指定`nodeProperties`。

类型：NodeProperties 对象

必需：否

`mainNode`

指定多节点并行作业的主节点的节点索引。此节点索引值必须小于节点数。

类型：整数

必需：是

`numNodes`

与多节点并行任务关联的节点数。

类型：整数

必需：是

`nodeRangeProperties`

与多节点并行作业关联的节点范围及其属性的列表。

> **Note**
>
> 节点组即共享相同容器属性的一组完全相同的作业节点。您可以使用 AWS Batch 为每个作业指定最多五个不同的节点组。

类型：NodeRange属性对象数组

必需：是

`targetNodes`

节点范围（使用节点索引值）。0:3范围表示索引值为 0 到 3 的节点。如果省略起始范围值 (:n)，则使用 0 开始范围。如果省略结束范围值 (n:)，则使用可能的最高节点索引结束范围。累积节点范围必须考虑所有节点 (0:n)。可以嵌套节点范围，例如0:10和4:5。在这种情况下，4:5范围属性会覆盖0:10属性。

类型：字符串

必需：否

`container`

节点范围的容器详细信息。有关更多信息，请参阅 容器属性。

类型：ContainerProperties 对象

必需：否

重试策略

`retryStrategy`

在注册作业定义时，针对使用此作业定义提交的失败作业，可以选择性地指定要用于这些作业的重试策略。在SubmitJob操作期间指定的任何重试策略都会覆盖此处定义的重试策略。默认情况下，每个作业尝试一次。如果指定多次尝试，则在作业失败的情况下，则会重试该作业。失败尝试的示例包括：作业返回非零退出代码或容器实例终止。有关更多信息，请参阅 自动作业重试。

类型：RetryStrategy 对象

必需：否

`attempts`

让作业进入RUNNABLE状态的次数。可以指定 1 到 10 之间的尝试次数。如果attempts大于 1，则当作业失败时将重试多次，直到它进入RUNNABLE状态。

```
"attempts": integer
```

类型：整数

必需：否

`evaluateOnExit`

最多由 5 个对象组成的数组，指定了在哪些条件下重试作业或作业失败。如果指定了此参数，则还必须指定`attempts`参数。如果指定了`evaluateOnExit`但没有匹配的条目，则会重试该作业。

```
"evaluateOnExit": [
   {
      "action": "string",
      "onExitCode": "string",
      "onReason": "string",
      "onStatusReason": "string"
   }
]
```

类型：EvaluateOn退出对象数组

必需：否

`action`

指定在满足所有指定条件（`onStatusReason`、`onReason`和`onExitCode`）时要执行的操作。这些值不区分大小写。

类型：字符串

必需：是

有效值：`RETRY` | `EXIT`

`onExitCode`

包含 glob 模式，用于与作业返回的`ExitCode`十进制表示法进行匹配。模式最多可包含 512 个字符。其中只能包含数字。不能包含字母或特殊字符。可以选择以星号 (*) 结束，这样只有字符串的开头需要完全匹配。

类型：字符串

必需：否

onReason

包含 glob 模式，用于与作业返回的Reason进行匹配。模式最多可包含 512 个字符。可以包含字母、数字、句点 (.)、冒号 (:) 和空格（空格、制表符）。可以选择以星号 (*) 结束，这样只有字符串的开头需要完全匹配。

类型：字符串

必需：否

onStatusReason

包含 glob 模式，用于与作业返回的StatusReason进行匹配。模式最多可包含 512 个字符。可以包含字母、数字、句点 (.)、冒号 (:) 和空格（空格、制表符）。可以选择以星号 (*) 结束，这样只有字符串的开头需要完全匹配。

类型：字符串

必需：否

计划优先级

schedulingPriority

使用此作业定义提交的作业的计划优先级。此项仅影响具有公平份额策略的作业队列中的作业。具有较高计划优先级的作业在具有较低计划优先级的作业之前计划。

支持的最小值为 0，支持的最大值为 9999。

类型：整数

必需：否

Tags

tags

与作业定义关联的键值配对标签。有关更多信息，请参阅 对AWS Batch资源加标签。

类型：字符串到字符串映射

必需：否

Timeout

`timeout`

您可以为作业配置超时时间，这样，如果作业运行的时间超过该时间，则 AWS Batch 终止该作业。有关更多信息，请参阅 作业超时。如果作业是因超时而终止，则不会重试。在SubmitJob操作期间指定的任何超时配置都会覆盖此处定义的超时配置。有关更多信息，请参阅 作业超时。

类型：JobTimeout 对象

必需：否

`attemptDurationSeconds`

`startedAt`终止未完成作业后的持续时间（从作业尝试的 AWS Batch 时间戳开始计算），以秒为单位。超时时间的最小值为 60 秒。

对于数组作业，超时适用于子作业，不适用于父数组作业。

对于多节点并行（MNP）作业，超时适用于整个作业，不适用于单个节点。

类型：整数

必需：否

使用创建作业定义 EcsProperties

使用 AWS Batch 作业定义后 `EcsProperties`，可以在单独的容器中对硬件、传感器、3D 环境和其他模拟进行建模。您可以使用此功能以逻辑方式组织工作负载组件，并将它们与主应用程序分开。此功能可在亚马逊弹性容器服务 (Amaz AWS Batch on ECS)、亚马逊 Elastic Kubernetes Service (Amazon EKS) 和 AWS Fargate

`ContainerProperties`与`EcsProperties`工作定义对比

您可以根据用例选择使用`ContainerProperties`或`EcsProperties`作业定义。简而言之，使用运行 AWS Batch 作业与使用`EcsProperties`运行作业类似`ContainerProperties`。

仍然支持使用`ContainerProperties`旧版作业定义结构。如果您当前有使用此结构的工作流，则可以继续运行它们。

主要区别在于，在作业定义中添加了一个新对象以适应`EcsProperties`基于的定义。

例如，`ContainerProperties`在 Amazon ECS 和 Fargate 上使用的任务定义具有以下结构：

```
{
   "containerProperties": {
     ...
     "image": "my_ecr_image1",
     ...
  },
...
}
```

EcsProperties在 Amazon ECS 和 Fargate 上使用的任务定义具有以下结构：

```
{
  "ecsProperties": {
    "taskProperties": [{
      "containers": [
        {
          ...
          "image": "my_ecr_image1",
          ...
        },
        {
          ...
          "image": "my_ecr_image2",
          ...
        },
```

对 AWS Batch API 的一般性更改

以下内容进一步概述了使用EcsProperties和 EcsProperties API 数据类型时的一些主要区别：

- 其中使用的许多参数都ContainerProperties出现在其中TaskContainerProperties。一些示例包括command、image、privileged、secrets、和users。它们都可以在里面找到TaskContainerProperties。
- 有些TaskContainerProperties参数在遗留结构中没有等效函数。一些示例包括dependsOn、essential、name、ipcMode、和pidMode。有关更多信息，请参阅EcsTaskDetails和TaskContainerProperties。

  而且，有些ContainerProperties参数在结构中没有等效项EcsProperties或应用。中taskProperties，container已替换为，containers因此新对象最多可以接受十个元素。有关更多信息，请参阅:容器属性和:容器RegisterJobDefinition。EcsTaskProperties

- taskRoleArn在功能上等同于。jobRoleArn有关更多信息，请参阅EcsTaskProperties:taskRoleArn和ContainerProperties:jobRoleArn。
- 可以在EcsProperties结构中包含一 (1) 到十 (10) 个容器。更多信息请参阅:容器EcsTaskProperties。
- taskProperties和 instanceTypes 对象是数组，但目前只接受一个元素。例如，: 任务属性和:EcsProperties实例类型。NodeRangeProperty

Amazon ECS 的多容器任务定义

为了适应 Amazon ECS 的多容器结构，某些 API 数据类型有所不同。例如，

- ecsProperties与单容器定义containerProperties中的级别相同。有关更多信息，请参阅 AWS Batch API 参考指南EcsProperties中的。
- taskProperties包含为 Amazon ECS 任务定义的属性。有关更多信息，请参阅 AWS Batch API 参考指南EcsProperties中的。
- containers包含与单容器定义containerProperties中类似的信息。主要区别在于，containers它允许您定义多达十个容器。有关更多信息，请参阅 API AWS Batch 参考指南TaskProperties中的 ECS: 容器。
- essential参数表示容器如何影响作业。所有 essential 容器都必须成功完成（以 0 退出），任务才能继续进行。如果标记为 essential 的容器失败（以非 0 退出），则作业将失败。

  默认值为，true并且必须至少将一个容器标记为essential。有关更多信息，请参阅 essential API 参考指南 中的 AWS Batch 。
- 使用dependsOn参数，您可以定义容器依赖关系的列表。有关更多信息，请参阅 dependsOn API 参考指南 中的 AWS Batch 。

> **Note**
>
> dependsOn列表的复杂性以及相关的容器运行时可能会影响任务的开始时间。如果依赖关系需要很长时间才能运行，则作业将保持STARTING状态，直到它们完成。

有关ecsProperties和结构的更多信息，请参阅 ecs Properties 的RegisterJobDefinition请求语法。

Amazon EKS 的多容器任务定义

为了适应 Amazon EKS 的多容器结构，某些 API 数据类型有所不同。例如，

- name是容器的唯一标识符。这个对象不是单个容器所必需的，但是在一个 pod 中定义多个容器时是必需的。如果name未为单个容器定义default，则应用默认名称。
- initContainers是在eksPodProperties数据类型中定义的。它们在应用程序容器之前运行，始终运行到完成，并且必须在下一个容器启动之前成功完成。

  这些容器已向 Amazon EKS Connector 代理注册，并将注册信息保存在亚马逊 Elastic Kubernetes Service 后端数据存储中。该initContainers对象最多可以接受十 (10) 个元素。有关更多信息，请参阅Kubernetes文档中的初始化容器。

  > **Note**
  >
  > 该initContainers对象可能会影响作业的开始时间。如果initContainers需要很长时间才能运行，则作业将一直处于STARTING状态，直到它们完成。

- shareProcessNamespace指示 Pod 中的容器是否可以共享相同的进程命名空间。默认值为false。将其设置为可以true让容器看到位于同一 pod 中的其他容器中的进程并发出信号。
- 每个容器都很重要。所有容器必须成功完成（以 0 退出），任务才能成功。如果一个容器失败（以 0 以外的身份退出），则作业将失败。

有关eksProperties和结构的更多信息，请参阅 eks Properties 的RegisterJobDefinition请求语法。

AWS Batch 使用的工作场景 EcsProperties

为了说明如何根据您的需求来构造所使用的 AWS Batch EcsProperties任务定义，本主题介绍了以下RegisterJobDefinition负载。您可以将这些示例复制到文件中，根据需要对其进行自定义，然后使用 AWS Command Line Interface (AWS CLI) 进行调用RegisterJobDefinition。

AWS Batch 亚马逊弹性容器服务在亚马逊弹性计算云上的工作

```
{
    "jobDefinitionName": "multicontainer-ecs-ec2",
    "type": "container",
    "ecsProperties": {
        "taskProperties": [
          {
            "containers": [
              {
                "name": "c1",
                "essential": false,
```

```
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "2"
                  },
                  {
                    "type": "MEMORY",
                    "value": "4096"
                  }
                ]
              },
              {
                "name": "c2",
                "essential": true,
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "6"
                  },
                  {
                    "type": "MEMORY",
                    "value": "12288"
                  }
                ]
              }
            ]
          }
        ]
  }
}
```

AWS Batch 在 Amazon ECS 上工作 AWS Fargate

```
{
    "jobDefinitionName": "multicontainer-ecs-fargate",
    "type": "container",
    "platformCapabilities": [
        "FARGATE"
    ],
    "ecsProperties": {
        "taskProperties": [
          {
            "containers": [
              {
                "name": "c1",
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "2"
                  },
                  {
                    "type": "MEMORY",
                    "value": "4096"
                  }
                ]
              },
              {
                "name": "c2",
                "essential": true,
                "command": [
                  "echo",
                  "hello world"
                ],
                "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                "resourceRequirements": [
                  {
                    "type": "VCPU",
                    "value": "6"
                  },
```

```
                {
                  "type": "MEMORY",
                  "value": "12288"
                }
              ]
            }
          ],
          "executionRoleArn": "arn:aws:iam::1112223333:role/ecsTaskExecutionRole"
        }
      ]
  }
}
```

AWS Batch 亚马逊 Elastic Kubernetes Service 的工作

```
{
  "jobDefinitionName": "multicontainer-eks",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "shareProcessNamespace": true,
      "initContainers": [
        {
          "name": "init-container",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "echo"
          ],
          "args": [
            "hello world"
          ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        },
        {
          "name": "init-container-2",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "echo",
```

```
            "my second init container"
          ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        }
      ],
      "containers": [
        {
          "name": "c1",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "echo world"
         ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        },
        {
          "name": "sleep-container",
          "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
          "command": [
            "sleep",
            "20"
          ],
          "resources": {
            "requests": {
              "cpu": "1",
              "memory": "512Mi"
            }
          }
        }
      ]
    }
  }
}
```

多节点并行 (MNP) AWS Batch 作业，每个节点有多个容器

```
{
  "jobDefinitionName": "multicontainer-mnp",
  "type": "multinode",
  "nodeProperties": {
    "numNodes": 6,
    "mainNode": 0,
    "nodeRangeProperties": [
      {
        "targetNodes": "0:5",
        "ecsProperties": {
          "taskProperties": [
            {
              "containers": [
                {
                  "name": "range05-c1",
                  "command": [
                    "echo",
                    "hello world"
                  ],
                  "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                  "resourceRequirements": [
                    {
                      "type": "VCPU",
                      "value": "2"
                    },
                    {
                      "type": "MEMORY",
                      "value": "4096"
                    }
                  ]
                },
                {
                  "name": "range05-c2",
                  "command": [
                    "echo",
                    "hello world"
                  ],
                  "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
                  "resourceRequirements": [
                    {
                      "type": "VCPU",
```

```
                    "value": "2"
                  },
                  {
                    "type": "MEMORY",
                    "value": "4096"
                  }
                ]
              }
            ]
          }
        ]
      }
    }
  ]
}
```

使用 awslogs 日志驱动程序

在默认情况下，AWS Batch将使awslogs日志驱动程序向 CloudWatch Logs 发送日志信息。可以使用此功能在同一个方便的位置查看容器中的不同日志，并防止容器日志占用容器实例上的磁盘空间。本主题可帮助在作业定义中配置awslogs日志驱动程序。

> **Note**
>
> 在AWS Batch控制台中，创建作业定义时，可以在awslogs日志记录配置部分配置日志驱动程序。

> **Note**
>
> 作业中的容器所记录的信息类型主要取决于其ENTRYPOINT命令。默认情况下，捕获的日志显示命令输出是在本地运行容器时在交互式终端上通常看到的内容，即STDOUT和STDERR I/O流。awslogs日志驱动程序只是将 Docker 中的这些日志传递到 CloudWatch Logs。有关如何处理 Docker 日志的更多信息，包括捕获不同文件数据或流的替代方法，请参阅 Docker 文档中的查看容器或服务的日志。

要将系统日志从容器实例发送到 CloudWatch Logs，请参阅将 CloudWatch 日志与配合使用 AWS Batch。有关 CloudWatch Logs 的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的监控日志文件和 CloudWatch Logs 配额。

可用的 awslogs 日志驱动程序选项

awslogs日志驱动程序支持AWS Batch作业定义中的下列选项。有关更多信息，请参阅 Docker 文档中的 CloudWatch 日志记录驱动程序。

awslogs-region

必需：否

指定awslogs日志驱动程序应将 Docker 日志发送到的区域。默认情况下，使用的区域与作业的区域相同。可以选择将来自不同区域作业的所有日志发送到 CloudWatch Logs 中的单个区域。这样，所有这些信息在同一个位置可见。或者，也可以按区域将它们分开，以获得更高粒度。但是，选择此选项时，请确保指定的日志组存在于指定的区域中。

awslogs-group

必需：可选

可以使用awslogs-group选项，以指定awslogs日志驱动程序将其日志流发送到的日志组。如果未指定，则将使用aws/batch/job。

awslogs-stream-prefix

必需：可选

通过awslogs-stream-prefix选项，可以将日志流与指定的前缀，以及容器所属AWS Batch作业的 Amazon ECS 任务 ID 关联在一起。如果使用此选项指定前缀，则日志流将采用以下格式：

```
prefix-name/default/ecs-task-id
```

awslogs-datetime-format

必需：否

此选项以 Python strftime 格式定义多行开始位置模式。日志消息由与模式匹配的行以及与模式不匹配的任何以下行组成。因此，匹配行是日志消息之间的分隔符。

使用此格式的一个使用案例示例是用于解析输出（如堆栈转储），这可能记录在多个条目中。正确模式允许它捕获在单个条目中。

有关更多信息，请参阅 awslogs-datetime-format。

如果同时配置了awslogs-datetime-format和awslogs-multiline-pattern，此选项始终优先。

> **Note**
>
> 多行日志记录对所有日志消息执行正则表达式解析和匹配。这可能会对日志记录性能产生负面影响。

awslogs-multiline-pattern

必需：否

此选项使用正则表达式定义多行开始位置模式。日志消息由与模式匹配的行以及与模式不匹配的任何以下行组成。因此，匹配行是日志消息之间的分隔符。

有关更多信息，请参阅 Docker 文档中的 awslogs-multiline-pattern。

如果还配置了awslogs-datetime-format，则会忽略此选项。

> **Note**
>
> 多行日志记录对所有日志消息执行正则表达式解析和匹配。这可能会对日志记录性能产生负面影响。

awslogs-create-group

必需：否

指定是否要自动创建日志组。如果未指定此选项，则默认为false。

> **Warning**
>
> 不建议使用该选项。建议使用 CloudW atch Logs CreateLogGroup API 操作提前创建日志组，因为每个作业都会尝试创建日志组，增加作业失败的机会。

> **Note**
>
> 执行角色的 IAM policy 必须包含`logs:CreateLogGroup`权限，然后才能尝试使用`awslogs-create-group`。

在作业定义中指定日志配置

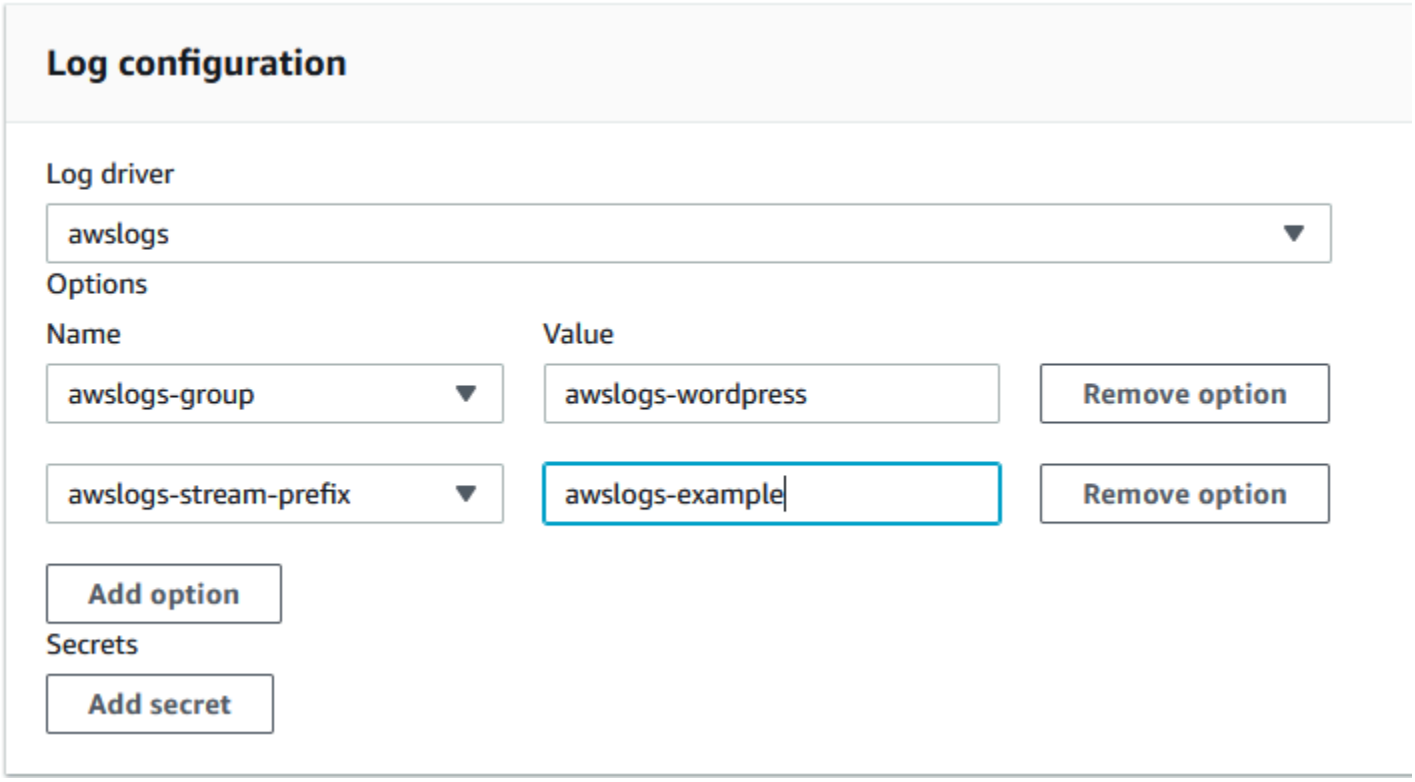
默认情况下，AWS Batch会启用`awslogs`日志驱动程序。本节介绍如何为作业自定义`awslogs`日志配置。有关更多信息，请参阅创建单节点作业定义 。

以下日志配置 JSON 片段为每个作业指定了一个`logConfiguration`对象。一个是用于将日志发送到名为`awslogs-wordpress`的日志组的 WordPress 作业，另一个是用于将日志发送到名为`awslogs-mysql`的日志组的 MySQL 容器。两个容器都使用`awslogs-example`日志流前缀。

```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "awslogs-wordpress",
        "awslogs-stream-prefix": "awslogs-example"
    }
}
```

```
"logConfiguration": {
    "logDriver": "awslogs",
    "options": {
        "awslogs-group": "awslogs-mysql",
        "awslogs-stream-prefix": "awslogs-example"
    }
}
```

AWS Batch控制台中指定了`wordpress`作业的日志配置，如以下映像所示。



在作业定义日志配置中向awslogs日志驱动程序注册作业定义之后，可以使用该作业定义提交作业，以开始将日志发送到 CloudWatch Logs。有关更多信息，请参阅提交作业。

指定敏感数据

借助 AWS Batch，您可以将敏感数据存储在 AWS Secrets Manager 密钥中或存储在 AWS Systems Manager Parameter Store 参数中，然后在作业定义中引用这些它们，从而将敏感数据注入作业。

可以按以下方式将密钥对作业开放：

- 要将敏感数据作为环境变量注入容器，请使用 `secrets` 作业定义参数。
- 要引用作业的日志配置中的敏感信息，请使用 `secretOptions` 作业定义参数。

主题

- 使用 Secrets Manager 密钥指定敏感数据
- 使用 Systems Manager Parameter Store 指定敏感数据

使用 Secrets Manager 密钥指定敏感数据

使用 AWS Batch，您可以将敏感数据注入作业，方法是将敏感数据存储在 AWS Secrets Manager 机密中，然后在作业定义中引用它们。存储在 Secrets Manager 密钥中的敏感数据可以作为环境变量或作为日志配置的一部分提供给作业。

在将密钥注入为环境变量时，可以指定 JSON 密钥或要注入的密钥的版本。此过程将帮助您控制提供给作业的敏感数据。有关密钥版本控制的更多信息，请参阅《AWS Secrets Manager 用户指南》中的AWS Secrets Manager的主要术语和概念。

有关使用 Secrets Manager 指定敏感数据的注意事项

在使用 Secrets Manager 指定作业的敏感数据时，应考虑以下事项。

- 要使用特定的 JSON 密钥或密钥版本注入密钥，您的计算环境中的容器实例必须安装版本 1.37.0 或更高版本的 Amazon ECS 容器代理。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本和更新到最新版本的信息，请参阅 Amazon Elastic Container Service 开发人员指南 中的 更新 Amazon ECS 容器代理。

  要将密钥的全部内容作为环境变量注入或在日志配置中注入密钥，您的容器实例必须有 1.23.0 或更高版本的容器代理。
- 仅支持存储文本数据的密钥，即使用 CreateSecretAPI 的`SecretString`参数创建的密钥。不支持存储二进制数据的密钥，即使用 CreateSecretAPI 的`SecretBinary`参数创建的密钥。
- 当使用引用 Secrets Manager 密钥的作业定义以检索作业的敏感数据时，如果您还在使用接口 VPC 端点，则必须为 Secrets Manager 创建接口 VPC 端点。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的将 Secrets Manager 与 VPC 端点结合使用。
- 最初启动作业时，会将敏感数据注入作业中。如果随后更新或轮换密钥，则作业将不会自动接收更新后的值。您必须启动一个新作业，才能强制服务启动具有更新的密钥值的新作业。

AWS Batch 密钥所需的 IAM 权限

要使用此功能，您必须具有作业执行角色，并在作业定义中引用它。这允许容器代理提取必要的 Secrets Manager 资源。有关更多信息，请参阅 AWS Batch 执行 IAM 角色。

要提供对您创建的 Secrets Manager 密钥的访问权限，请将以下权限作为内联策略手动添加到执行角色。有关更多信息，请参阅 IAM 用户指南中的添加和删除 IAM 策略。

- `secretsmanager:GetSecretValue`– 引用 Secrets Manager 密钥时的必填项。

- `kms:Decrypt` - 仅当您的密钥使用自定义 KMS 密钥而不是原定设置密钥时才需要。您的自定义密钥的 ARN 应添加为资源。

以下示例内联策略会添加所需权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:<secret_name>",
        "arn:aws:kms:<region>:<aws_account_id>:key/<key_id>"
      ]
    }
  ]
}
```

作为环境变量注入敏感数据

在作业定义中，可以指定以下项目：

- `secrets` 对象包含要在作业中设置的环境变量的名称
- Secrets Manager 密钥的 Amazon 资源名称（ARN）
- 包含要提供给作业的敏感数据的其他参数

以下示例显示必须为 Secrets Manager 密钥指定的完整语法。

```
arn:aws:secretsmanager:region:aws_account_id:secret:secret-name:json-key:version-stage:version-id
```

下一部分介绍了其他参数。这些参数是可选的。但如果您不使用它们，则必须包含冒号 : 以使用默认值。下面提供了示例，以便您了解更多上下文。

`json-key`

使用要设置为环境变量值的值指定密钥-值对中密钥的名称。仅支持 JSON 格式的值。如果未指定 JSON 密钥，则使用密钥的完整内容。

`version-stage`

指定要使用的密钥版本的暂存标签。如果指定了版本的暂存标签，则无法指定版本 ID。如果未指定版本阶段，则默认行为是使用 AWSCURRENT 暂存标签检索密钥。

暂存标签用于在更新或轮换密钥的各个版本时对其进行跟踪。密钥的每个版本均有一个或多个暂存标签和一个 ID。有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的 S AWS ecrets Manager 的关键术语和概念。

`version-id`

指定要使用的密钥版本的唯一标识符。如果指定了版本 ID，则无法指定版本暂存标签。如果未指定版本 ID，则默认行为是使用 AWSCURRENT 暂存标签检索密钥。

版本 ID 用于在更新或轮换密钥的各个版本时对其进行跟踪。密码的每个版本均有一个 ID。有关更多信息，请参阅《AWS Secrets Manager 用户指南》中的 S AWS ecrets Manager 的关键术语和概念。

示例容器定义

以下示例说明了可用于在容器定义中引用 Secrets Manager 密钥的方法。

Example 引用完整密钥

以下是任务定义的片段，其中显示引用 Secrets Manager 密钥的完全文本时的格式。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
    }]
  }]
}
```

Example 引用密钥中的特定密钥

下面显示了一个命令的输出示例，该get-secret-value命令显示了密钥的内容以及与之关联的版本暂存标签和版本 ID。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "VersionId": "871d9eca-18aa-46a9-8785-981dd39ab30c",
    "SecretString": "{\"username1\":\"password1\",\"username2\":\"password2\",
\"username3\":\"password3\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": 1581968848.921
}
```

通过在 ARN 的末尾指定密钥名称，引用容器定义中的上一个输出中的特定密钥。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::"
    }]
  }]
}
```

Example 引用特定的密钥版本

下面显示了 describe-secret 命令中的示例输出，其中显示了密钥的未加密内容以及密钥的所有版本的元数据。

```
{
    "ARN": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-AbCdEf",
    "Name": "appauthexample",
    "Description": "Example of a secret containing application authorization data.",
    "RotationEnabled": false,
    "LastChangedDate": 1581968848.926,
    "LastAccessedDate": 1581897600.0,
    "Tags": [],
```

```
    "VersionIdsToStages": {
        "871d9eca-18aa-46a9-8785-981dd39ab30c": [
            "AWSCURRENT"
        ],
        "9d4cb84b-ad69-40c0-a0ab-cead36b967e8": [
            "AWSPREVIOUS"
        ]
    }
}
```

通过在 ARN 的末尾指定密钥名称，引用容器定义中的上一个输出中的特定版本暂存标签。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::AWSPREVIOUS:"
    }]
  }]
}
```

通过在 ARN 的末尾指定密钥名称，引用容器定义中的上一个输出中的特定版本 ID。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf::9d4cb84b-ad69-40c0-a0ab-cead36b967e8"
    }]
  }]
}
```

Example 引用密钥的特定密钥和版本暂存标签

以下内容说明如何同时引用密钥中的特定密钥和特定版本暂存标签。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
```

```
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1:AWSPREVIOUS:"
    }]
  }]
}
```

要指定特定密钥和版本 ID，请使用以下语法。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:appauthexample-
AbCdEf:username1::9d4cb84b-ad69-40c0-a0ab-cead36b967e8"
    }]
  }]
}
```

注入日志配置中的敏感数据

在作业定义中，当指定 `logConfiguration` 时，您可以同时指定 `secretOptions`，方法是使用要在容器中设置的日志驱动程序选项的名称，以及包含要提供给容器的敏感数据的 Secrets Manager 密钥的完整 ARN。

以下是作业定义的片段，其中显示引用 Secrets Manager 密钥时的格式。

```
{
  "containerProperties": [{
    "logConfiguration": [{
      "logDriver": "splunk",
      "options": {
        "splunk-url": "https://cloud.splunk.com:8080"
      },
      "secretOptions": [{
        "name": "splunk-token",
        "valueFrom": "arn:aws:secretsmanager:region:aws_account_id:secret:secret_name-
AbCdEf"
      }]
    }]
  }]
}
```

创建密 AWS Secrets Manager 钥

您可以使用 Secrets Manager 控制台为您的敏感数据创建密钥。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的创建基本密钥。

创建基本密钥

使用 Secrets Manager 为您的敏感数据创建密钥。

1. 在 https://console.aws.amazon.com/secretsmanager/ 打开 Secrets Manager 控制台。
2. 选择 存储新密钥。
3. 对于选择密钥类型，选择其他密钥类型。
4. 以 Key (键) 和 Value (值) 对的形式指定自定义密钥的详细信息。例如，您可以指定键 `UserName`，然后提供适当的用户名作为其值。添加名为 `Password` 的第二个键并将密码文本作为其值。您还可以为数据库名称、服务器地址、TCP 端口等添加条目。您可以添加所需数量的对以存储所需的信息。

   或者，您也可以选择 Plaintext (明文) 选项卡，并以所需的任何方式输入密钥值。
5. 选择要用于 AWS KMS 加密密钥中受保护文本的加密密钥。如果您没有选择一个，则 Secrets Manager 会检查账户是否存在原定设置密钥并在存在时使用它。如果不存在原定设置密钥，则 Secrets Manager 将自动为您创建一个。您也可以选择添加新密钥以创建专用于该密钥的自定义 KMS 键。要创建您自己的 KMS 键，您必须具有在您的账户中创建 KMS 键的权限。
6. 选择下一步。
7. 对于密钥名称，请键入可选的路径和名称，如 **`production/MyAwesomeAppSecret`** 或 **`development/TestSecret`**，然后选择下一步。您可以选择添加描述以帮助记住该密钥以后的用途。

   密钥名称应仅包含 ASCII 字母、数字或以下任意字符：/_+=.@-
8. (可选) 此时，您可以为密钥配置轮换。对于此程序，请将其保留为禁用自动轮换，然后选择下一步。

   有关如何配置新密钥或现有密钥的轮换的信息，请参阅轮换您的 AWS Secrets Manager 密钥。
9. 检查您的设置，然后选择 Store secret (存储密钥) 以将输入的所有内容作为新密钥保存在 Secrets Manager 中。

使用 Systems Manager Parameter Store 指定敏感数据

使用 AWS Batch，您可以将敏感数据注入容器，方法是将敏感数据存储在 P AWS Systems Manager arameter Store 参数中，然后在容器定义中引用它们。

主题

使用 Systems Manager Parameter Store 指定敏感数据时的注意事项

使用 Systems Manager Parameter Store 参数指定容器的敏感数据时，应考虑以下事项。

- 此功能要求您的容器实例具有 1.23.0 或更高版本的容器代理。但是，我们建议使用最新的容器代理版本。有关检查您的代理版本和更新到最新版本的信息，请参阅 Amazon Elastic Container Service 开发人员指南 中的 更新 Amazon ECS 容器代理。
- 最初启动容器时，会为您的作业将敏感数据注入容器中。如果随后更新或轮换密钥或 Parameter Store 参数，则容器将不会自动接收已更新的值。您必须启动一个新作业，才能强制启动一个包含更新的密钥的新作业。

AWS Batch 密钥所需的 IAM 权限

要使用此功能，您必须具有作业执行角色，并在作业定义中引用它。这允许 Amazon ECS 容器代理提取必要的 AWS Systems Manager 资源。有关更多信息，请参阅 AWS Batch 执行 IAM 角色。

要提供对您创建的 P AWS Systems Manager arameter Store 参数的访问权限，请手动将以下权限作为内联策略添加到执行角色。有关更多信息，请参阅 IAM 用户指南中的添加和删除 IAM 策略。

- `ssm:GetParameters`— 当您在任务定义中引用 Systems Manager Parameter Store 参数时，这是必填项。
- `secretsmanager:GetSecretValue`— 当您直接引用 Secrets Manager 密钥或者您的 System Manager Parameter Store 参数在任务定义中引用 Secrets Manager 密钥时，这是必填项。

- `kms:Decrypt`- 仅当您的密钥使用自定义 KMS 密钥而不是默认密钥时才需要。您的自定义密钥的 ARN 应添加为资源。

以下示例内联策略添加所需权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters",
        "secretsmanager:GetSecretValue",
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:ssm:<region>:<aws_account_id>:parameter/<parameter_name>",
        "arn:aws:secretsmanager:<region>:<aws_account_id>:secret:<secret_name>",
        "arn:aws:kms:<region>:<aws_account_id>:key/<key_id>"
      ]
    }
  ]
}
```

作为环境变量注入敏感数据

在容器定义中，使用要在容器中设置的环境变量的名称和包含要提供给容器的敏感数据的 Systems Manager Parameter Store 参数的完整 ARN 指定 `secrets`。

以下是任务定义的片段，其中显示引用 Systems Manager Parameter Store 参数时的格式。如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用参数的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

```
{
  "containerProperties": [{
    "secrets": [{
      "name": "environment_variable_name",
      "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
    }]
  }]
}
```

注入日志配置中的敏感数据

在容器定义中，当指定 `logConfiguration` 时，您可以使用要在容器中设置的日志驱动程序选项的名称以及包含要提供给容器的敏感数据的 Systems Manager Parameter Store 参数的完整 ARN 指定 `secretOptions`。

> **Important**
>
> 如果 Systems Manager Parameter Store 参数存在于要启动的任务所在的区域，则可以使用参数的完整 ARN 或名称。如果参数存在于不同的区域，则必须指定完整的 ARN。

以下是任务定义的片段，其中显示引用 Systems Manager Parameter Store 参数时的格式。

```
{
  "containerProperties": [{
    "logConfiguration": [{
      "logDriver": "fluentd",
      "options": {
        "tag": "fluentd demo"
      },
      "secretOptions": [{
        "name": "fluentd-address",
        "valueFrom": "arn:aws:ssm:region:aws_account_id:parameter/parameter_name"
      }]
    }]
  }]
}
```

创建 AWS Systems Manager 参数存储参数

您可以使用 AWS Systems Manager 控制台为敏感数据创建 Systems Manager 参数存储参数。有关更多信息，请参见 AWS Systems Manager 用户指南 中的 演练：在命令（控制台）中创建和使用参数。

创建 Parameter Store 参数

1. 打开 AWS Systems Manager 控制台，网址为 https://console.aws.amazon.com/systems-manager/。
2. 在导航窗格中，依次选择 Parameter Store 和 Create parameter (创建参数)。
3. 对于 Name (名称)，键入层次结构和参数名称。例如，键入 `test/database_password`。

4. 对于 Description (描述)，键入可选描述。
5. 对于 “类型”，选择 “字符串” StringList、“或” SecureString。

> **Note**
>
> - 如果您选择 SecureString，则会出现 KMS 密钥 ID 字段。如果您没有提供 KMS 密钥 ID、KMS 密钥 ARN、别名或别名 ARN，则系统将使用 `alias/aws/ssm`。这是 Systems Manager 的默认 KMS 密钥。要避免使用此密钥，请选择自定义密钥。有关安全字符串的更多信息，请参阅 AWS Systems Manager 用户指南中的使用安全字符串参数。
> - 在控制台中使用具有自定义 KMS 键 别名或别名 ARN 的 `key-id` 参数创建安全字符串参数时，您必须在别名前面指定前缀 `alias/`。以下是 ARN 示例：
>
>   ```
>   arn:aws:kms:us-east-2:123456789012:alias/MyAliasName
>   ```
>
>   以下是别名示例：
>
>   ```
>   alias/MyAliasName
>   ```

6. 对于 Value (值)，键入一个值。例如，`MyFirstParameter`。如果您选择 SecureString，则该值将完全按照您输入的值进行屏蔽。
7. 选择创建参数。

作业的私有注册表身份验证

使用对作业进行私有注册表身份验证 AWS Secrets Manager 使您能够安全地存储凭证，然后在作业定义中引用这些凭证。这提供了一种引用私有注册表中存在的容器镜像的方法 AWS ，这些镜像需要在任务定义中进行身份验证。托管在 Amazon EC2 实例和 Fargate 上的作业支持此功能。

> **Important**
>
> 如果您的任务定义引用了存储在 Amazon ECR 中的图片，则本主题不适用。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的使用 Amazon ECR 和 Amazon ECS。

对于托管在 Amazon EC2 实例上的任务，此功能需要容器代理版本`1.19.0`或更高版本。但是，我们建议使用最新的容器代理版本。有关如何检查代理版本和更新到最新版本的信息，请参阅《亚马逊弹性容器服务开发者指南》中的更新 Amazon ECS 容器代理。

对于托管在 Fargate 上的作业，此功能需要平台版本`1.2.0`或更高版本。有关信息，请参阅《亚马逊弹性容器服务开发者指南》中的 AWS Fargate Linux 平台版本。

在容器定义中，使用您创建的密钥的详细信息指定 `repositoryCredentials` 对象。你引用的密钥可以来自与使用它的工作不同的账户， AWS 区域 也可以来自不同的账户。

> **Note**
>
> 使用 AWS Batch API AWS CLI、或 AWS SDK 时，如果密钥与您启动的任务 AWS 区域 相同，则可以使用密钥的完整 ARN 或名称。如果密钥存在于另一个账户中，则必须指定密钥的完整 ARN。使用时 AWS Management Console，必须始终指定密钥的完整 ARN。

以下是显示所需参数的作业定义片段：

```
"containerProperties": [
  {
    "image": "private-repo/private-image",
    "repositoryCredentials": {
      "credentialsParameter":
 "arn:aws:secretsmanager:region:123456789012:secret:secret_name"
    }
  }
]
```

私有注册表身份验证所需的 IAM 权限

需要执行角色才能使用此功能。这允许容器代理拉取容器映像。有关更多信息，请参阅AWS Batch 执行 IAM 角色。

要提供对您创建的密钥的访问权限，请将以下权限作为内联策略添加到执行角色。有关更多信息，请参阅添加和删除 IAM policy。

- `secretsmanager:GetSecretValue`
- `kms:Decrypt` - 仅当密钥使用自定义 KMS 密钥而不是原定设置密钥时才需要。您的自定义密钥的 Amazon 资源名称（ARN）必须添加为资源。

下面是添加所需权限的示例内联策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:region:123456789012:secret:secret_name",
        "arn:aws:kms:region:123456789012:key/key_id"
      ]
    }
  ]
}
```

使用私有注册表身份验证

创建基本密钥

AWS Secrets Manager 用于为您的私有注册表凭证创建密钥。

1. 打开 AWS Secrets Manager 控制台，网址为 https://console.aws.amazon.com/secretsmanager/。
2. 选择 存储新密钥。
3. 对于选择密钥类型，选择其他密钥类型。
4. 选择纯文本文件并使用以下格式输入您的私有注册表凭证：

   ```
   {
     "username" : "privateRegistryUsername",
     "password" : "privateRegistryPassword"
   }
   ```

5. 选择下一步。
6. 对于 Secret name（密钥名称），请输入可选的路径和名称，如 **production/MyAwesomeAppSecret** 或 **development/TestSecret**，然后选择 Next（下一步）。您可以选择添加描述以帮助记住该密钥以后的用途。

密钥名称应仅包含 ASCII 字母、数字或以下任意字符：/_+=.@-。

7. (可选) 此时，您可以为密钥配置轮换。对于此程序，请将其保留为禁用自动轮换，然后选择下一步。

   有关如何配置新密钥或现有密钥轮换的说明，请参阅轮换您的 AWS Secrets Manager 密钥。

8. 检查您的设置，然后选择 Store secret（存储密钥）以将输入的所有内容作为新密钥保存在 Secrets Manager 中。

注册作业定义，然后在 “私有注册表” 下打开 “私有注册表身份验证”。然后，在 Secrets Manager ARN 或名称中，输入密钥的 Amazon 资源名称（ARN）。有关更多信息，请参阅 私有注册表身份验证所需的 IAM 权限。

Amazon EFS 卷

Amazon Elastic File System (Amazon EFS) 提供简单的可扩展文件存储以供AWS Batch作业使用。使用 Amazon EFS 时，存储容量是弹性的。它会随着添加和删除文件而自动扩展。应用程序可在需要时获得所需存储。

可以将 Amazon EFS 文件系统与AWS Batch配合使用，以便导出跨容器实例的实例集的文件系统数据。这样，作业就可以访问相同的永久存储。但是，必须将容器实例 AMI 配置为在 Docker 进程守护程序启动前挂载 Amazon EFS 文件系统。此外，作业定义必须引用容器实例上的卷挂载才能使用该文件系统。下面几个部分可帮助开始使用 Amazon EFS 与AWS Batch配合使用。

Amazon EFS 卷注意事项

使用 Amazon EFS 卷时应注意以下事项：

- 对于使用 EC2 资源的作业，已将 Amazon EFS 文件系统支持作为公开预览版添加，其中包括 Amazon ECS 优化 AMI 版本20191212以及容器代理版本 1.35.0。但是，Amazon EFS 文件系统支持通过 Amazon ECS 优化 AMI 版本20200319和容器代理版本 1.38.0 正式推出，该版本包含 Amazon EFS 接入点和 IAM 授权功能。建议使用 Amazon ECS 优化 AMI 版本20200319或更高版本以利用这些功能。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 优化 AMI 版本。

> **Note**
>
> 如果创建自己的 AMI，则必须使用容器代理 1.38.0 或更高版本、`ecs-init`版本 1.38.0-1 或更高版本，并在 Amazon EC2 实例上运行以下命令。这一切都是为了启用 Amazon ECS 卷插件。命令取决于将 Amazon Linux 2 还是 Amazon Linux 用作基本映像。
>
> Amazon Linux 2
>
> ```
> $ yum install amazon-efs-utils
> systemctl enable --now amazon-ecs-volume-plugin
> ```
>
> Amazon Linux
>
> ```
> $ yum install amazon-efs-utils
> sudo shutdown -r now
> ```

- 对于 Fargate 资源的作业，使用 1.4.0 或更高的平台版本时，添加了 Amazon EFS 文件系统支持。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的AWS Fargate 平台版本。
- 在使用 Fargate 资源的作业中指定 Amazon EFS 卷时，Fargate 会创建负责管理 Amazon EFS 卷的主管容器。主管容器使用少量的作业内存。主管容器在查询任务元数据版本 4 端点时可见。有关更多信息，请参阅《Amazon Elastic Container Service AWS Fargate用户指南》中的任务元数据端点版本 4。

使用 Amazon EFS 接入点

Amazon EFS 接入点是 EFS 文件系统中特定于应用程序的入口点，便于轻松地管理应用程序对共享数据集的访问。有关 Amazon EFS 接入点以及如何控制访问的更多信息，请参阅《Amazon Elastic File System 用户指南》中的使用 Amazon EFS 接入点。

接入点可以为通过接入点发出的所有文件系统请求强制执行用户身份（包括用户的 POSIX 组）。接入点还可以为文件系统强制执行不同的根目录，以便客户端只能访问指定目录或其子目录中的数据。

> **Note**
>
> 创建 EFS 接入点时，可以在文件系统上指定用作根目录的路径。在AWS Batch作业定义中引用具有接入点 ID 的 EFS 文件系统时，必须忽略根目录或将根目录设置为/，以便在 EFS 接入点上强制执行设置的路径。

可以使用AWS Batch作业 IAM 角色强制特定应用程序使用某个具体的接入点。可以通过将 IAM 策略与接入点相结合轻松地为应用程序提供对特定数据集的安全访问。此功能使用 Amazon ECS IAM 角色来执行任务功能。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的任务的 IAM 角色。

在作业定义中指定 Amazon EFS 文件系统

要为容器使用 Amazon EFS 文件系统卷，必须在作业定义中指定卷和挂载点配置。以下作业定义 JSON 代码段显示容器的`volumes`和`mountPoints`对象的语法：

```
{
    "containerProperties": [
        {
            "image": "amazonlinux:2",
            "command": [
                "ls",
                "-la",
                "/mount/efs"
            ],
            "mountPoints": [
                {
                    "sourceVolume": "myEfsVolume",
                    "containerPath": "/mount/efs",
                    "readOnly": true
                }
            ],
            "volumes": [
                {
                    "name": "myEfsVolume",
                    "efsVolumeConfiguration": {
                        "fileSystemId": "fs-12345678",
                        "rootDirectory": "/path/to/my/data",
                        "transitEncryption": "ENABLED",
                        "transitEncryptionPort": integer,
```

```
                        "authorizationConfig": {
                            "accessPointId": "fsap-1234567890abcdef1",
                            "iam": "ENABLED"
                        }
                    }
                }
            ]
        }
    ]
}
```

`efsVolumeConfiguration`

类型：对象

必需：否

使用 Amazon EFS 卷时将指定此参数。

`fileSystemId`

类型：字符串

必需：是

要使用的 Amazon EFS 文件系统 ID。

`rootDirectory`

类型：字符串

必需：否

Amazon EFS 文件系统中要作为主机内的根目录挂载的目录。如果忽略此参数，将使用 Amazon EFS 卷的根目录。指定/与忽略此参数效果相同。其长度最多为 4096 个字符。

> **Important**
>
> 如果在`authorizationConfig`中指定了 EFS 接入点，则必须省略根目录参数，或者将其设置为/。这将强制执行 EFS 接入点上设置的路径。

`transitEncryption`

类型：字符串

有效值：ENABLED | DISABLED

必需：否

确定是否对AWS Batch主机和 Amazon EFS 服务器之间传输的 Amazon EFS 数据启用加密。如果使用 Amazon EFS IAM 授权，则必须启用传输加密。如果忽略此参数，将使用默认值DISABLED。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的加密传输中数据。

`transitEncryptionPort`

类型：整数

必需：否

在AWS Batch主机和 Amazon EFS 服务器之间发送加密数据时要使用的端口。如果未指定传输加密端口，将使用 Amazon EFS 挂载帮助程序使用的端口选择策略。该值必须在 0 到 65535 之间。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的 EFS 挂载帮助程序。

`authorizationConfig`

类型：对象

必需：否

Amazon EFS 文件系统的授权配置详细信息。

`accessPointId`

类型：字符串

必需：否

要使用的接入点 ID。如果指定了接入点，则必须省略在`efsVolumeConfiguration`中的根目录值，或者将其设置为`/`。这将强制执行 EFS 接入点上设置的路径。如果使用接入点，则必须在`EFSVolumeConfiguration`中启用传输加密。有关更多信息，请参阅《Amazon Elastic File System 用户指南》中的使用 Amazon EFS 接入点。

`iam`

类型：字符串

有效值：ENABLED | DISABLED

必需：否

确定在挂载 Amazon EFS 文件系统时是否使用在作业定义中定义的AWS Batch作业 IAM 角色。如果启用，则必须在EFSVolumeConfiguration中启用传输加密。如果忽略此参数，将使用默认值DISABLED。有关执行 IAM 角色的更多信息，请参阅AWS Batch 执行 IAM 角色。

作业定义示例

以下作业定义示例说明了如何使用环境变量、参数替换和卷挂载等常用模式。

使用环境变量

以下作业定义示例使用环境变量来指定文件类型和 Amazon S3 URL。该特定示例来自创建简单的“Fetch & Run”AWS Batch 作业计算博客文章。博客文章中描述的fetch_and_run.sh脚本使用这些环境变量从 S3 下载myjob.sh脚本并声明其文件类型。

尽管在本示例中，命令和环境变量被硬编码到作业定义中，但仍可指定命令和环境变量覆盖，使作业定义更具通用性。

```
{
    "jobDefinitionName": "fetch_and_run",
    "type": "container",
    "containerProperties": {
        "image": "123456789012.dkr.ecr.us-east-1.amazonaws.com/fetch_and_run",
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "2000"
            },
            {
                "type": "VCPU",
                "value": "2"
            }
        ],
        "command": [
            "myjob.sh",
            "60"
        ],
        "jobRoleArn": "arn:aws:iam::123456789012:role/AWSBatchS3ReadOnly",
        "environment": [
```

```
            {
                "name": "BATCH_FILE_S3_URL",
                "value": "s3://my-batch-scripts/myjob.sh"
            },
            {
                "name": "BATCH_FILE_TYPE",
                "value": "script"
            }
        ],
        "user": "nobody"
    }
}
```

使用参数替代

以下示例作业定义说明了如何允许参数替代和设置默认值。

Ref::一节中的command声明用于设置参数替代的占位符。提交使用此作业定义的作业时，可以指定参数覆盖以填充这些值，例如inputfile和outputfile。下面的parameters一节设置了codec默认值，但可以根据需要覆盖该参数。

有关更多信息，请参阅参数。

```
{
    "jobDefinitionName": "ffmpeg_parameters",
    "type": "container",
    "parameters": {"codec": "mp4"},
    "containerProperties": {
        "image": "my_repo/ffmpeg",
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "2000"
            },
            {
                "type": "VCPU",
                "value": "2"
            }
        ],
        "command": [
            "ffmpeg",
            "-i",
            "Ref::inputfile",
```

```
            "-c",
            "Ref::codec",
            "-o",
            "Ref::outputfile"
        ],
        "jobRoleArn": "arn:aws:iam::123456789012:role/ECSTask-S3FullAccess",
        "user": "nobody"
    }
}
```

测试 GPU 功能

在以下示例中，作业定义测试使用 GPU 工作负载 AMI中所述的 GPU 工作负载 AMI 是否正确配置。此示例作业定义运行来自 GitHub 的 Tensorflow deep MNIST 分类器示例。

```
{
    "containerProperties": {
        "image": "tensorflow/tensorflow:1.8.0-devel-gpu",
        "resourceRequirements": [
            {
                "type": "MEMORY",
                "value": "32000"
            },
            {
                "type": "VCPU",
                "value": "8"
            }
        ],
        "command": [
            "sh",
            "-c",
            "cd /tensorflow/tensorflow/examples/tutorials/mnist; python mnist_deep.py"
        ]
    },
    "type": "container",
    "jobDefinitionName": "tensorflow_mnist_deep"
}
```

可以创建名为`tensorflow_mnist_deep.json`的文件来包含上面的 JSON 文本，然后使用以下命令注册AWS Batch作业定义：

```
aws batch register-job-definition --cli-input-json file://tensorflow_mnist_deep.json
```

多节点并行作业

以下作业定义示例描述了多节点并行作业。有关更多信息，请参阅AWS计算博客中的使用多节点并行作业构建紧密耦合AWS Batch的分子动力学工作流程。

```
{
  "jobDefinitionName": "gromacs-jobdef",
  "jobDefinitionArn": "arn:aws:batch:us-east-2:123456789012:job-definition/gromacs-
jobdef:1",
  "revision": 6,
  "status": "ACTIVE",
  "type": "multinode",
  "parameters": {},
  "nodeProperties": {
    "numNodes": 2,
    "mainNode": 0,
    "nodeRangeProperties": [
      {
        "targetNodes": "0:1",
        "container": {
          "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/gromacs_mpi:latest",
          "resourceRequirements": [
              {
                  "type": "MEMORY",
                  "value": "24000"
              },
              {
                  "type": "VCPU",
                  "value": "8"
              }
          ],
          "command": [],
          "jobRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
          "ulimits": [],
          "instanceType": "p3.2xlarge"
        }
      }
    ]
  }
}
```

作业队列

作业将提交到作业队列，并将一直存放在队列中，直到能够在计算环境中计划运行这些作业。一个 AWS 账户可以有多个任务队列。例如，可以为高优先级作业创建一个使用 Amazon EC2 按需型实例的队列，为低优先级作业创建另一个使用 Amazon EC2 竞价型实例的队列。作业队列具有优先级，调度器使用该优先级来确定应评估哪个队列中的作业首先执行。

主题

创建作业队列

在您可在 AWS Batch 中提交作业之前，必须先创建一个作业队列。在创建作业队列时，您可以将一个或多个计算环境与队列相关联，并且分配优先顺序。

您还可以为作业队列设置优先级，该队列决定了 AWS 批处理调度器放置作业的顺序。这意味着，如果计算环境与多个作业队列关联，则具有较高优先级的作业队列将会优先作业。

创建 Fargate 作业队列

要创建 Fargate 作业队列

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的 AWS 区域。
3. 在导航窗格中，选择 作业队列。
4. 选择 Create（创建）。
5. 对于编排类型，请选择 Fargate。
6. 对于名称，为作业队列输入唯一名称。名称可以长达 128 个字符，可以包含大小写字母、数字及下划线 (_)。
7. 对于优先级，为作业队列的优先级输入一个整数值。具有较高优先级的作业队列在与同一计算环境关联的较低优先级作业队列之前运行。优先级按降序确定。例如，优先级值为 10 的任务队列将会比优先级值为 1 的任务队列优先计划。

8. （可选）对于计划策略 Amazon 资源名称（ARN），请选择现有的计划策略。
9. 对于 连接的计算环境，从列表中选择一个或多个计算环境，以便与作业队列关联。按照您希望队列尝试放置作业队列的顺序选择计算环境。作业计划程序使用您选择的计算环境顺序来确定哪些计算环境应启动给定作业。计算环境必须先处于 VALID 状态，然后您才能将其与作业队列关联。您最多可以将三个计算环境与一个作业队列关联。

   > **Note**
   >
   > 与作业队列关联的所有计算环境必须共享同一配置模型。AWS Batch 不支持在单个作业队列中混合使用配置模型。

10. 对于计算环境顺序，选择向上和向下箭头以配置所需的顺序。
11. 选择 创建作业队列 以完成和创建作业队列。

创建 Amazon EC2 任务队列

要创建 Amazon EC2 作业队列

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的 AWS 区域。
3. 在导航窗格中，选择 作业队列。
4. 选择 Create（创建）。
5. 对于编排类型，选择 Amazon Elastic Compute Cloud (Amazon EC2)。
6. 对于名称，为作业队列输入唯一名称。名称可以长达 128 个字符，可以包含大小写字母、数字及下划线 (_)。
7. 对于优先级，为作业队列的优先级输入一个整数值。具有较高优先级的作业队列在与同一计算环境关联的较低优先级作业队列之前运行。优先级按降序确定。例如，优先级值为 10 的任务队列将会比优先级值为 1 的任务队列优先计划。
8. （可选）对于计划策略 Amazon 资源名称（ARN），请选择现有的计划策略。
9. 对于 连接的计算环境，从列表中选择一个或多个计算环境，以便与作业队列关联。按照您希望队列尝试放置作业队列的顺序选择计算环境。作业计划程序使用您选择的计算环境顺序来确定哪些计算环境应启动给定作业。计算环境必须先处于 VALID 状态，然后您才能将其与作业队列关联。您最多可以将三个计算环境与一个作业队列关联。如果您没有现有的计算环境，请选择创建计算环境

> **Note**
>
> 与作业队列关联的所有计算环境必须共享同一配置模型。AWS Batch 不支持在单个作业队列中混合使用配置模型。

10. 对于计算环境顺序，选择向上和向下箭头以配置所需的顺序。
11. 选择 创建作业队列 以完成和创建作业队列。

创建 Amazon EKS 作业队列

要创建 Amazon EKS 作业 队列

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的 AWS 区域。
3. 在导航窗格中，选择 作业队列。
4. 选择 Create（创建）。
5. 对于编排类型，选择 Amazon Elastic Kubernetes Service (Amazon EKS)。
6. 对于名称，为作业队列输入唯一名称。名称可以长达 128 个字符，可以包含大小写字母、数字及下划线 (_)。
7. 对于 Priority，为作业队列的优先级输入一个整数值。具有较高优先级的作业队列在与同一计算环境关联的较低优先级作业队列之前运行。优先级按降序确定。例如，优先级值为 10 的任务队列将会比优先级值为 1 的任务队列优先计划。
8. （可选）对于计划策略 Amazon 资源名称（ARN），请选择现有的计划策略。
9. 对于 连接的计算环境，从列表中选择一个或多个计算环境，以便与作业队列关联。按照您希望队列尝试放置作业队列的顺序选择计算环境。作业计划程序使用您选择的计算环境顺序来确定哪些计算环境应启动给定作业。计算环境必须先处于 `VALID` 状态，然后您才能将其与作业队列关联。您最多可以将三个计算环境与一个作业队列关联。

> **Note**
>
> 与作业队列关联的所有计算环境必须共享同一配置模型。AWS Batch 不支持在单个作业队列中混合使用配置模型。

> **Note**
>
> 与作业队列关联的所有计算环境必须共享同一架构。AWS Batch 不支持在单个作业队列中混合使用计算环境架构类型。

10. 对于计算环境顺序，选择向上和向下箭头以配置所需的顺序。
11. 选择 创建作业队列 以完成和创建作业队列。

作业队列模板

以下是一个空的作业队列模板。可以使用此模板创建作业队列。然后，您可以将此作业队列保存到文件中，然后将其与 AWS CLI `--cli-input-json`选项一起使用。有关这些参数的更多信息，请参阅 AWS Batch API 参考CreateJobQueue中的。

```
{
   "computeEnvironmentOrder": [
      {
         "computeEnvironment": "",
         "order": 0
      }
   ],
   "jobQueueName": "",
   "jobStateTimeLimitActions": [
      {
         "state": "RUNNABLE",
         "action": "CANCEL",
   "maxTimeSeconds": 0,
         "reason": ""

      }
   ],
   "priority": 0,
   "schedulingPolicyArn": "",
   "state": "ENABLED",
   "tags": {
      "KeyName": ""
   }
}
```

> **Note**
>
> 您可以使用以下 AWS CLI 命令生成前面的作业队列模板。
>
> ```
> $ aws batch create-job-queue --generate-cli-skeleton
> ```

作业队列参数

Job 队列分为四个基本组成部分：名称、状态、优先级和计算环境顺序。本节讨论与这些组件相关的组件。

主题

作业队列名称

`jobQueueName`

作业队列的名称。允许使用不超过 128 个字母 (大写和小写字母)、数字和下划线。

类型：字符串

必需：是

Job 队列状态时间限制操作

jobStateTimeLimitActions

对处于指定状态且停留在作业队列开头的时间超过指定时间的作业 AWS Batch 执行的一组操作。AWS Batch 将在通过后maxTimeSeconds执行每个操作。（注意：的最小值maxTimeSeconds为 600（10 分钟），其最大值为 86,400（24 小时）。）

类型：JobStateTimeLimitActions 对象数组

必需：否

优先级

priority

作业队列的优先级。当有多个作业队列与同一计算环境关联时，系统将首先评估具有较高优先级（或priority参数的较高整数值）的作业队列。优先级按降序顺序确定，例如，优先级值为 10 的作业队列将会比优先级值为 1 的作业队列优先计划。所有计算环境都必须是 Amazon EC2（EC2或SPOT）或 Fargate（FARGATE或FARGATE_SPOT）。Amazon EC2 和 Fargate 的计算环境不能混合使用。

类型：整数

必需：是

计划策略

schedulingPolicyArn

作业队列计划策略的 Amazon 资源名称（ARN）。没有计划策略的作业队列采用先进先出 (FIFO) 模式进行计划。作业队列采用计划策略后，可以替换该队列，但不能将其删除。没有计划策略的作业队列是作为 FIFO 作业队列计划的，不能添加计划策略。采用计划策略的作业队列最多可以有 500 个有效公平份额标识符。达到限制后，任何添加新的公平份额标识符的作业提交都将失败。

类型：字符串

必需：否

省/自治区/直辖市

state

作业队列的状态。如果作业队列状态为ENABLED（默认值），它可以接受作业。如果作业队列状态为DISABLED，则无法将新作业添加到队列中，但队列中已有的作业可以完成。

类型：字符串

有效值：ENABLED | DISABLED

必需：否

计算环境顺序

computeEnvironmentOrder

一组计算环境，它们映射到一个作业队列，并且其顺序是彼此相关的。作业调度器使用此参数来确定哪个计算环境应执行指定作业。计算环境必须先处于VALID状态，然后才能将其与作业队列关联。最多可以将三个计算环境与一个作业队列关联。所有计算环境都必须是 Amazon EC2（EC2或SPOT）或 Fargate（FARGATE或FARGATE_SPOT）。Amazon EC2 和 Fargate 的计算环境不能混合使用。

> **Note**
>
> 与任务队列关联的所有计算环境都必须共享相同的架构。 AWS Batch 不支持在单个作业队列中混合计算环境架构类型。

类型：ComputeEnvironmentOrder 对象数组

必需：是

computeEnvironment

计算环境的 Amazon 资源名称（ARN）。

类型：字符串

必需：是

`order`

计算环境的顺序。计算环境按照升序顺序尝试。例如，如果有两个计算环境与一个作业队列关联，则具有较低`order`整数值的计算环境首先尝试进行作业置放。

标签

`tags`

与作业队列关联的键值配对标签。有关更多信息，请参阅 对AWS Batch资源加标签。

类型：字符串到字符串映射

必需：否

查看作业队列状态

创建任务队列并提交作业后，能够监控其进度非常重要。您可以使用 Job 详情页面来查看、管理和监控您的作业队列。

查看作业队列信息

在 AWS Batch 控制台中，在导航窗格中选择 J ob queues，然后选择所需的任务队列以查看其详细信息。在此页面上，您可以查看和管理您的任务队列并查看有关队列操作的其他信息，例如作业队列快照、作业状态限制、环境顺序、标签和作业队列的 JSON 代码。

Job 队列详情

本节提供作业队列的概述和维护选项。请务必注意，您可以在本节中找到亚马逊资源名称 (ARN)。

要通过查找此信息 AWS Command Line Interface，请使用`DescribeJobQueues`操作以及任务队列名称或相应的 ARN。

Job 队列快照

本节提供队列中前 100 个`RUNNABLE`作业的静态列表。您可以使用搜索字段通过搜索结果区任意列中的信息来缩小列表范围。快照结果区域中的作业根据作业队列的运行策略进行排序。对于 first-in-first-out (FIFO) 任务队列，作业的排序基于提交时间。对于AWS Batch 公平共享计划 (FSS) 作业队列，作业的排序基于作业优先级和份额使用情况。

由于结果是作业队列的快照，因此结果列表不会自动更新。要更新列表，请选择该部分顶部的刷新。选择作业的名称超链接可导航至任务详细信息并查看该作业的状态和其他相关信息。

要通过查找此信息 AWS CLI，请使用GetJobQueueSnapshot操作以及任务队列名称或相应的 ARN。

Job 状态限制

使用此选项卡可以查看有关任务在取消之前可以保持RUNNABLE状态的时间的配置信息。

要通过查找此信息 AWS CLI，请使用DescribeJobQueues操作以及任务队列名称或相应的 ARN。

环境顺序

如果您的任务队列在多个环境中运行，则此选项卡会提供它们的顺序和概述。

要通过查找此信息 AWS CLI，请使用DescribeJobQueues操作以及任务队列名称或相应的 ARN。

标签

使用此选项卡可以查看和管理与此作业队列关联的标签。

JSON

使用此选项卡复制与此作业队列关联的 JSON 代码。然后，您可以将 JSON 重复用于 AWS CloudFormation 模板和 AWS CLI 脚本。

任务计划

AWS Batch 调度器评估何时、何地以及如何运行提交到作业队列的作业。如果您在创建作业队列时未指定调度策略，则 AWS Batch 作业调度器默认为先进先出 (FIFO) 策略。FIFO 策略可能会导致重要的工作“滞留”在之前提交的工作之后。通过指定不同的调度策略，您可以根据自己的特定需求分配计算资源。

> **Note**
>
> 如果要安排作业的特定运行顺序，请使用中的dependsOn参数SubmitJob来指定每个作业的依赖关系。

如果您创建调度策略并将其附加到作业队列，则公平份额调度将处于启用状态。如果作业队列有调度策略，则调度策略决定作业的运行顺序。有关更多信息，请参阅 计划策略。

份额标识符

您可以使用份额标识符来标记作业并区分用户和工作负载。 AWS Batch 调度器使用公*(T * weightFactor)*式跟踪每个公平份额标识符的使用情况，其中*T*是随时间变化的 vCPU 使用情况。调度器从份额标识符中挑选使用率最低的作业。您可以使用公平份额标识符而不将其覆盖。

> **Note**
>
> 份额标识符在作业队列中是唯一的，并且不会在作业队列中汇总。

您可以设置调度优先级，根据份额标识符配置作业的运行顺序。具有较高调度优先级的作业优先调度。如果您未指定调度策略，则所有提交到作业队列的作业都将按照 FIFO 顺序进行调度。提交作业时，您不能指定份额标识符或计划优先级。

> **Note**
>
> 除非明确覆盖，否则附加的计算资源将在所有份额标识符之间平均分配。

公平份额调度

公平份额调度提供了一组控件来帮助安排作业。

> **Note**
>
> 有关调度策略参数的更多信息，请参阅 计划策略参数。

- 股票衰减秒 — AWS Batch 调度程序用于计算每个公平份额标识符的公平份额百分比的时间段（以秒为单位）。值为零表示仅测量当前使用量。更长的衰减时间会增加时间的权重。

  > **Note**
  >
  > 衰减时间段的计算公式为：*shareDecaySeconds + OrderMinutes* 其中 *OrderMinutes* 是顺序中的时间（以分钟为单位）。

- 计算预留 - 防止单个份额标识符中的作业耗尽附加到作业队列的所有资源。预留比率是*computeReservation/100)^ActiveFairShares*活跃的公平份额标识符的数量。ActiveFairShares

  > **Note**
  >
  > 如果份额标识符的作业处于SUBMITTED、PENDING、RUNNABLE、STARTING 或 RUNNING 状态，则该标识符被视为有效份额标识符。衰减期限到期后，份额标识符被视为非活动状态。

- 权重系数 - 份额标识符的权重系数。默认值是 1。较低的值允许份额标识符中的作业运行，或者为份额标识符提供额外的运行时间。例如，使用权重因子为 0.125（1/8）的共享标识符的作业获得的计算资源是使用权重因子为 1 的共享标识符的作业的 8 倍。

  > **Note**
  >
  > 只有在需要更新默认权重系数 1 时才需要定义此属性。

当作业队列处于活动状态并正在处理作业时，您可以通过作业队列快照查看RUNNABLE前 100 个作业的列表。有关更多信息，请参阅查看作业队列状态。

计算环境

作业队列映射到一个或多个计算环境。计算环境包含用于运行容器化批处理作业的 Amazon ECS 容器实例。还可以将一个具体的计算环境映射到一个或多个作业队列。在作业队列中，每个关联的计算环境均有一个顺序，计划程序使用该顺序来确定准备好运行的作业将在哪里运行。如果第一个计算环境的状态为VALID且有可用资源，作业就会被调度到该计算环境中的容器实例。如果第一个计算环境的状态为INVALID 或无法提供合适的计算资源，计划程序会尝试在下一个计算环境上运行任务。

主题

托管计算环境

您可以使用托管计算环境来 AWS Batch 管理环境中计算资源的容量和实例类型。这基于在创建计算环境时定义的计算资源规范。可以选择使用 Amazon EC2 按需型实例和 Amazon EC2 竞价型实例。或者，也可以在托管计算环境中使用 Fargate 和 Fargate 竞价容量。使用竞价型实例时，可以选择设置最高价格。这样，只有当竞价型实例价格低于按需型实例价格的指定百分比时，竞价型实例才会启动。

> **Important**
>
> 不支持 Fargate Spot 实例。 Windows containers on AWS Fargate如果将作业提交到仅使用 Fargate Spot 计算环境的作业队列，则任务队列将被阻止。 FargateWindows

托管计算环境会在指定的 VPC 和子网中启动 Amazon EC2 实例，然后将其注册到 Amazon ECS 集群。Amazon EC2 实例需要外部网络访问权限才能与 Amazon ECS 服务端点通信。有些子网不为 Amazon EC2 实例提供公共 IP 地址。如果您的 Amazon EC2 实例没有公共 IP 地址，必须使用网络地址转换（NAT）才能获得访问权。有关更多信息，请参阅《Amazon VPC 用户指南》中的 NAT 网关。有关如何创建 VPC 的更多信息，请参阅创建虚拟私有云 。

默认情况下， AWS Batch 托管计算环境使用最新经批准的 Amazon ECS 优化版 AMI 来处理计算资源。但是，可能出于各种原因需要创建自己的 AMI，用于托管计算环境。有关更多信息，请参阅 计算资源 &AMI;。

> **Note**
>
> AWS Batch 创建计算环境中的 AMI 后，不会自动对其进行升级。例如，当更新版本的 Amazon ECS 优化 AMI 发布时，它不会更新计算环境中的 AMI。您需要管理客户操作系统。其中包括任何更新和安全补丁。您还负责为在计算资源上安装的任何其他应用程序软件或实用程序。有两种方法可以将新 AMI 用于您的 AWS Batch 工作。最初的方法是完成以下步骤：
>
> 1. 使用新的 AMI 创建新计算环境。
> 2. 将计算环境添加到现有作业队列。
> 3. 从作业队列中删除早期的计算环境。
> 4. 删除早期的计算环境。
>
> 2022 年 4 月， AWS Batch 增加了对更新计算环境的增强支持。有关更多信息，请参阅 更新计算环境。要使用计算环境的增强更新来更新 AMI，请遵循以下规则：
>
> - 要么不要设置服务角色 (`serviceRole`) 参数，要么将其设置为AWSServiceRoleForBatch服务相关角色。
> - 将分配策略 (`allocationStrategy`) 参数设置为BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED或SPOT_PRICE_CAPACITY_OPTIMI
> - 将更新到最新图像版本(`updateToLatestImageVersion`)参数设置为true。
> - 请勿在`imageId`、`imageIdOverride`（在`ec2Configuration`）或启动模板 (`launchTemplate`) 中指定 AMI ID。在这种情况下，请 AWS Batch 选择基础设施更新启动时支持的最新 Amazon ECS 优化的 AMI。 AWS Batch 或者，可以在imageId或imageIdOverride参数中指定 AMI ID，也可以在LaunchTemplate属性中指定启动模板。更改这些属性中的任何一个都将启动基础架构更新。如果在启动模板中指定了 AMI ID，则不能通过在imageId或imageIdOverride参数中指定 AMI ID 来替换它。只能通过指定不同的启动模板来替换它。或者如果启动模板版本设置

为$Default或$Latest，则为启动模板设置新的默认版本（如果是$Default），或者为启动模板添加新版本（如果是$Latest）。

如果遵循这些规则，任何启动基础架构更新的更新都会导致重新选择 AMI ID。如果启动模板 (launchTemplate) 中的version设置设为$Latest或$Default，则在基础架构更新时会评估启动模板的最新版本或默认版本，即使launchTemplate尚未更新。

创建多节点并行作业时的注意事项

AWS Batch 建议创建专用的计算环境来运行多节点并行 (MNP) 作业和非 MNP 作业。这是由于在托管计算环境中创建计算容量的方式造成的。创建新的托管计算环境时，如果指定的minvCpu值大于零，则 AWS Batch 会创建一个仅供非 MNP 作业使用的实例池。如果提交了多节点并行作业，则 AWS Batch 会创建新的实例容量来运行多节点并行作业。如果在设置了minvCpus或maxvCpus值的同一个计算环境中同时运行单节点和多节点并行作业，则如果所需的计算资源不可用，则 AWS Batch 将等待当前作业完成，然后再创建运行新作业所需的计算资源。

非托管计算环境

在非托管计算环境中，需要管理自己的计算资源。必须验证用于计算资源的 AMI 是否符合 Amazon ECS 容器实例 AMI 规范。有关更多信息，请参阅 计算资源 &AMI; 规范 和 创建计算资源 &AMI;。

> **Note**
>
> AWS 非托管计算环境不支持 Fargate 资源。

创建非托管计算环境后，使用 DescribeComputeEnvironmentsAPI 操作查看计算环境的详细信息。找到与环境相关联的 Amazon ECS 集群，然后手动在该 Amazon ECS 集群中启动容器实例。

以下 AWS CLI 命令还提供了 Amazon ECS 集群 ARN。

```
$ aws batch describe-compute-environments \
    --compute-environments unmanagedCE \
    --query "computeEnvironments[].ecsClusterArn"
```

有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的启动 Amazon ECS 容器实例。启动计算资源时，请指定 Amazon ECS 集群 ARN，以便资源使用以下 Amazon EC2 用户数据注册。*ecsClusterArn*替换为您通过上一个命令获得的集群 ARN。

```
#!/bin/bash
echo "ECS_CLUSTER=ecsClusterArn" >> /etc/ecs/ecs.config
```

计算资源 &AMI;

默认情况下， AWS Batch 托管计算环境使用最新经批准的 Amazon ECS 优化版 AMI 来处理计算资源。但是，您可能出于以下原因需要创建自己的 &AMI; 以用于托管计算环境和非托管计算环境：如果您需要以下任何一项，我们建议您创建自己的 AMI：

- 增加 &AMI; 根卷或数据卷的存储大小
- 为支持的 Amazon EC2 实例类型添加实例存储卷
- 检查 Amazon ECS 容器代理
- 自定义 Docker
- 配置 GPU 工作负载 AMI，它允许容器访问支持的 Amazon EC2 实例类型上的 GPU 硬件

> **Note**
>
> 创建计算环境后， AWS Batch 不会升级计算环境中的 AMI。 AWS Batch 当有更新版本的 Amazon ECS 优化的 AMI 可用时，也不会更新您的计算环境中的 AMI。您需要管理客户操作系统。其中包括任何更新和安全补丁。您还负责为在计算资源上安装的任何其他应用程序软件或实用程序。要将新的 AMI 用于您的 AWS Batch 任务，请执行以下操作：
>
> 1. 使用新的 AMI 创建新计算环境。
> 2. 将计算环境添加到现有作业队列。
> 3. 从作业队列中删除早期的计算环境。
> 4. 删除早期的计算环境。
>
> 2022 年 4 月， AWS Batch 增加了对更新计算环境的增强支持。有关更多信息，请参阅 更新计算环境。要使用计算环境的增强更新来更新 AMI，请遵循以下规则：
>
> - 要么不要设置服务角色 (`serviceRole`) 参数，要么将其设置为AWSServiceRoleForBatch服务相关角色。

- 将分配策略（allocationStrategy）参数设置为 BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED 或 SPOT_PRICE_CAPACITY_OPTIMIZED。
- 将更新到最新图像版本(updateToLatestImageVersion)参数设置为true。
- 请勿在imageId、imageIdOverride（在ec2Configuration）或启动模板(launchTemplate) 中指定 AMI ID。如果您不指定 AMI ID，请 AWS Batch 选择在启动基础设施更新时 AWS Batch 支持的最新 Amazon ECS 优化的 AMI。或者，您可以指定 imageIdOverride 参数，而不是 imageId。或者，也可以指定由 LaunchTemplate 属性标识的启动模板。更改这些属性中的任何一个都将启动基础架构更新。如果在启动模板中指定了 AMI ID，则不能通过在 imageId 或 imageIdOverride 参数中指定 AMI ID 来替换它。AMI ID 只能通过指定不同的启动模板进行替换。如果启动模板版本设置为 $Default 或 $Latest，则可以通过为启动模板设置新的默认版本（如果 $Default）或向启动模板添加新版本（如果 $Latest）来替换 AMI ID。

如果遵循这些规则，触发基础设施更新的任何更新都将导致重新选择 AMI ID。如果启动模板(launchTemplate) 中的version设置设置为$Latest或$Default，则在基础架构更新时会评估启动模板的最新版本或默认版本，即使launchTemplate未更新。

主题

计算资源 &AMI; 规范

基本 AWS Batch 计算资源 AMI 规范包括以下内容：

必填

- 在 HVM 虚拟化类型 &AMI; 上运行至少 3.10 版 Linux 内核的现代 Linux 分配。不支持 Windows 容器。

> **Important**
>
> 多节点并行作业只能在安装了`ecs-init`程序包的 Amazon Linux 实例上启动的计算资源上运行。我们建议您在创建计算环境时使用默认的经过 Amazon ECS 优化的 AMI。您可以通过不指定自定义 AMI 来执行此操作。有关更多信息，请参阅 多节点并行作业。

- 停止 Amazon ECS 容器代理。建议您使用最新的 版本。有关更多信息，请参阅Amazon Elastic Container Service 开发人员指南中的安装 Amazon ECS 容器代理。
- 在启动 Amazon ECS 容器代理时，必须使用`ECS_AVAILABLE_LOGGING_DRIVERS`环境变量将`awslogs`日志驱动程序指定为可用的日志驱动程序。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 容器代理配置。
- 运行至少 1.9 版的 Docker 进程守护程序以及任何 Docker 运行时依赖项。有关更多信息，请参阅 Docker 文档中的检查运行时依赖项。

> **Note**
>
> 要获得最佳体验，建议您使用所使用的相应 Amazon ECS 容器 代理版本附带的且经测试的 Docker 版本。Amazon ECS 提供了针对亚马逊 ECS 优化的 AMI 的 Linux 变体的变更日志。 GitHub有关更多信息，请参阅更改日志。

推荐

- 用于运行和监控 Amazon ECS 容器 代理的初始化和 nanny 流程。经 Amazon ECS 优化的 AMI 使用 `ecs-init` upstart 流程，其他操作系统可能使用 `systemd`。有关更多信息和示例，请参阅 Amazon Elastic Container Service 开发人员指南中的示例容器实例用户数据配置脚本。有关的更多信息`ecs-init`，请参阅上的`ecs-init`项目 GitHub。托管计算环境至少需要 Amazon ECS 代理才能在系统启动时启动。如果 Amazon ECS 代理未在您的计算资源上运行，则它将无法接受来自的任务 AWS Batch。

经 Amazon ECS 优化的 AMI 已根据这些要求和建议进行了预配置。建议您将经 Amazon ECS 优化的 AMI 或 Amazon Linux AMI 与为您的计算资源安装的 `ecs-init`程序包一起使用。如果您的应用程序需要特定的操作系统或这些 AMI 中尚未提供的 Docker 版本，请选择另一个 AMI。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 经Amazon ECS 优化的 AMI。

创建计算资源 &AMI;

您可以创建您自己的自定义计算资源 AMI 以用于托管计算环境和非托管计算环境。有关说明，请参阅计算资源 &AMI; 规范。在创建自定义 AMI 后，您可以创建一个使用该 AMI 的计算环境，将此环境与一个任务队列关联，然后开始将任务提交到该队列。最后，开始向该队列提交作业。

创建自定义计算资源 &AMI;

1. 选择从中启动的基本 &AMI;。AMI 必须使用 HVM 虚拟化。基础 AMI 不能是 Windows AMI。

   > **Note**
   >
   > 您为计算环境选择的 AMI 必须与您希望用于该计算环境的实例类型的架构匹配。例如，如果您的计算环境使用 A1 实例类型，则您选择的计算资源 AMI 必须支持 Arm 实例。Amazon ECS 同时提供经过 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 Arm 版本。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的经过 Amazon ECS 优化的 Amazon Linux 2 AMI。

   经 Amazon ECS 优化的 Amazon Linux 2 AMI 是托管计算环境中的计算资源的默认 AMI。优化的亚马逊 ECS Amazon Linux 2 AMI 已 AWS Batch 由 AWS 工程师进行预配置和测试。这是一款最低限度的 AMI，您可以开始使用它并让您的计算资源 AWS 快速运行。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 经Amazon ECS 优化的 AMI。

   或者，您可以选择另一个 Amazon Linux 2 变体，并使用以下命令安装 `ecs-init` 程序包：有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的在 Amazon Linux 2 EC2 实例中安装 Amazon ECS 容器代理。

   ```
   $ sudo amazon-linux-extras disable docker
   $ sudo amazon-linux-extras install ecs-init
   ```

   例如，如果您想在 AWS Batch 计算资源上运行 GPU 工作负载，则可以从 Amazon Linux 深度学习 AMI 开始。然后，将 AMI 配置为运行 AWS Batch 作业。有关更多信息，请参阅 使用 GPU 工作负载 AMI。

> **Important**
>
> 您可以选择不支持 `ecs-init` 软件包的基础 AMI。但是，如果这样做，则必须配置一种在启动时启动 Amazon ECS 代理并使其保持运行的方法。您还可以查看几个使用 `systemd` 启动和监控 Amazon ECS 容器代理的用户数据配置脚本示例。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的示例容器实例用户数据配置脚本。

2. 使用适用于 &AMI; 的存储选项从选定的基本 &AMI; 启动实例。您可以配置附加的 Amazon EBS 卷或实例存储卷 (如果选定实例类型支持实例存储卷) 的大小和数量。有关更多信息，请参阅 Amazon EC2 用户指南中的启动实例和 Amazon EC2 实例存储。
3. 使用 SSH 连接到您的实例并执行任何必要的配置任务，例如：SSH这可能包括以下任一或所有步骤：

   - 安装 Amazon ECS 容器代理 有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的安装 Amazon ECS 容器代理。
   - 配置脚本以设置实例存储卷的格式。
   - 将实例存储卷或 Amazon EFS 文件系统添加到 `/etc/fstab` 文件，以便它们在系统启动时挂载。
   - 配置 Docker 选项 (启用调试、调整基本映像大小等)。
   - 安装程序包或复制文件。

   有关更多信息，请参阅 Amazon EC2 用户指南中的使用 SSH 连接到您的 Linux 实例。
4. 如果您在实例上启动了 Amazon ECS 容器代理，则在创建 AMI 之前，必须将其停止并移除所有永久性数据检查点文件。否则，如果您不这样做，代理不会在从您的 AMI 启动的实例上启动。

   a. 停止 Amazon ECS 容器代理。

      - 经 Amazon ECS 优化的 Amazon ECS Amazon Linux 2 AMI：

        ```
        sudo systemctl stop ecs
        ```

      - 经 Amazon ECS 优化的 Amazon ECS Amazon Linux AMI：

        ```
        sudo stop ecs
        ```

b. 删除持久性数据检查点文件。默认情况下，该文件位于以下 `/var/lib/ecs/data/` 目录中。使用以下命令删除这些文件（如果有）。

```
sudo rm -rf /var/lib/ecs/data/*
```

5. 从正在运行的实例创建新的 &AMI;。有关更多信息，请参阅亚马逊 EC2 用户指南指南中的创建由亚马逊 EBS 支持的 Linux AMI。

要将您的新 AMI 与 AWS Batch

1. 使用新的 AMI 创建新计算环境。要做到这一点， 选择镜像类型并在镜像 ID 中输入自定义 AMI ID 创建 AWS Batch 计算环境时覆盖框。有关更多信息，请参阅 the section called “若要使用 EC2 资源创建托管计算环境”。

> **Note**
>
> 您为计算环境选择的 AMI 必须与您希望用于该计算环境的实例类型的架构匹配。例如，如果您的计算环境使用 A1 实例类型，则您选择的计算资源 AMI 必须支持 Arm 实例。Amazon ECS 同时提供经过 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 Arm 版本。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的经过 Amazon ECS 优化的 Amazon Linux 2 AMI。

2. 创建作业队列并关联新计算环境。有关更多信息，请参阅 创建作业队列。

> **Note**
>
> 与任务队列关联的所有计算环境都必须共享相同的架构。 AWS Batch 不支持在单个作业队列中混合计算环境架构类型。

3. （可选）将示例作业提交到新作业队列。有关更多信息，请参阅 作业定义示例、创建单节点作业定义 和 提交作业。

使用 GPU 工作负载 AMI

要在您的 AWS Batch 计算资源上运行 GPU 工作负载，必须使用具有 GPU 支持的 AMI。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的在 Amazon ECS 上使用 GPU 以及 Amazon ECS 优化的 AMI。

在托管计算环境中，如果计算环境指定任何 p2, p3, p4, p5, g3, g3s, g4 或 g5 实例类型或实例系列，则 AWS Batch 使用经过 Amazon ECS GPU 优化的 AMI。

在非托管计算环境中，建议使用经过 Amazon ECS GPU 优化的 AMI。可以使用 AWS Command Line Interface 或 AWS Systems Manager Parameter Store GetParameter、GetParameters 和 GetParametersByPath 操作来检索元数据，从而获得建议的经过 Amazon ECS GPU 优化的 AMI。

> **Note**
>
> 只有等于或高于 Amazon ECS GPU 优化的 AMI 20230912 的版本才支持 p5 实例系列，并且它们与 p2 和 g2 实例类型不兼容。如果您需要使用 p5 实例，请确保您的计算环境不包含 p2 或 g2 实例，并使用最新的默认 Batch AMI。创建新的计算环境将使用最新的 AMI，但是如果您要更新计算环境以包含 p5，则可以通过在 `ComputeResource` 属性中将 `updateToLatestImageVersion` 设置为 `true` 来确保使用的是最新的 AMI。有关 AMI 与 GPU 实例兼容的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的在 Amazon ECS 上使用 GPU。

以下示例演示了如何使用 GetParameter 命令。

AWS CLI

```
$ aws ssm get-parameter --name /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended \
                        --region us-east-2 --output json
```

输出在 `Value` 参数中包含 AMI 信息。

```
{
    "Parameter": {
        "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended",
        "LastModifiedDate": 1555434128.664,
        "Value": "{\"schema_version\":1,\"image_name\":\"amzn2-ami-ecs-gpu-
hvm-2.0.20190402-x86_64-ebs\",\"image_id\":\"ami-083c800fe4211192f\",\"os\":\"Amazon
 Linux 2\",\"ecs_runtime_version\":\"Docker version 18.06.1-ce\",\"ecs_agent_version
\":\"1.27.0\"}",
        "Version": 9,
        "Type": "String",
        "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended"
```

```
    }
}
```

Python

```
from __future__ import print_function

import json
import boto3

ssm = boto3.client('ssm', 'us-east-2')

response = ssm.get_parameter(Name='/aws/service/ecs/optimized-ami/amazon-linux-2/
gpu/recommended')
jsonVal = json.loads(response['Parameter']['Value'])
print("image_id   = " + jsonVal['image_id'])
print("image_name = " + jsonVal['image_name'])
```

输出仅包含 AMI ID 和 AMI 名称：

```
image_id   = ami-083c800fe4211192f
image_name = amzn2-ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs
```

以下示例演示 GetParameters 的用法。

AWS CLI

```
$ aws ssm get-parameters --names  /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended/image_name \
                                  /aws/service/ecs/optimized-ami/amazon-linux-2/gpu/
recommended/image_id \
                          --region us-east-2 --output json
```

输出包含每个参数的完整元数据：

```
{
    "InvalidParameters": [],
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_id",
```

```
            "LastModifiedDate": 1555434128.749,
            "Value": "ami-083c800fe4211192f",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_id"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_name",
            "LastModifiedDate": 1555434128.712,
            "Value": "amzn2-ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_name"
        }
    ]
}
```

Python

```
from __future__ import print_function

import boto3

ssm = boto3.client('ssm', 'us-east-2')

response = ssm.get_parameters(
            Names=['/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_name',
                   '/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_id'])
for parameter in response['Parameters']:
    print(parameter['Name'] + " = " + parameter['Value'])
```

输出包含 AMI ID 和 AMI 名称，并使用名称的完整路径。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_id =
 ami-083c800fe4211192f
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_name = amzn2-
ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs
```

以下示例显示如何使用 GetParametersByPath 命令。

AWS CLI

```
$ aws ssm get-parameters-by-path --path /aws/service/ecs/optimized-ami/amazon-
linux-2/gpu/recommended \
                                  --region us-east-2 --output json
```

输出包含指定路径下的所有参数的完整元数据。

```
{
    "Parameters": [
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
ecs_agent_version",
            "LastModifiedDate": 1555434128.801,
            "Value": "1.27.0",
            "Version": 8,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/ecs_agent_version"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
ecs_runtime_version",
            "LastModifiedDate": 1548368308.213,
            "Value": "Docker version 18.06.1-ce",
            "Version": 1,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/ecs_runtime_version"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_id",
            "LastModifiedDate": 1555434128.749,
            "Value": "ami-083c800fe4211192f",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_id"
        },
        {
```

```
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
image_name",
            "LastModifiedDate": 1555434128.712,
            "Value": "amzn2-ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs",
            "Version": 9,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/image_name"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
os",
            "LastModifiedDate": 1548368308.143,
            "Value": "Amazon Linux 2",
            "Version": 1,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/os"
        },
        {
            "Name": "/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/
schema_version",
            "LastModifiedDate": 1548368307.914,
            "Value": "1",
            "Version": 1,
            "Type": "String",
            "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ecs/optimized-ami/
amazon-linux-2/gpu/recommended/schema_version"
        }
    ]
}
```

Python

```
from __future__ import print_function

import boto3

ssm = boto3.client('ssm', 'us-east-2')

response = ssm.get_parameters_by_path(Path='/aws/service/ecs/optimized-ami/amazon-
linux-2/gpu/recommended')
for parameter in response['Parameters']:
```

```
    print(parameter['Name'] + " = " + parameter['Value'])
```

输出包含指定路径下的所有参数名称的值，使用完整路径名称。

```
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/ecs_agent_version =
 1.27.0
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/ecs_runtime_version =
 Docker version 18.06.1-ce
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_id =
 ami-083c800fe4211192f
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/image_name = amzn2-
ami-ecs-gpu-hvm-2.0.20190402-x86_64-ebs
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/os = Amazon Linux 2
/aws/service/ecs/optimized-ami/amazon-linux-2/gpu/recommended/schema_version = 1
```

有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的检索 Amazon ECS 优化的 AMI 元数据。

Amazon Linux 弃用

亚马逊 Linux AMI（也称为亚马逊 Linux 1）于 2023 年 12 月 31 日停产。 AWS Batch 已终止对亚马逊 Linux AMI 的支持，因为它从 2024 年 1 月 1 日起将不会收到任何安全更新或错误修复。有关 Amazon Linux 的更多信息 end-of-life，请参阅 A L 常见问题解答。

我们建议您将基于 Amazon Linux 的现有计算环境更新到 Amazon Linux 2023，以防止不可预见的工作负载中断，并继续接收安全和其他更新。

您使用亚马逊 Linux AMI 的计算环境可能会在 2023 年 end-of-life 12 月 31 日之后继续运行。但是，这些计算环境将不再收到来自的任何新软件更新、安全补丁或错误修复 AWS。之后，您有责任在 Amazon Linux AMI 上维护这些计算环境 end-of-life。我们建议将 AWS Batch 计算环境迁移到亚马逊 Linux 2023 或亚马逊 Linux 2，以保持最佳性能和安全性。

要获得 AWS Batch 从亚马逊 Linux AMI 迁移到亚马逊 Linux 2023 或亚马逊 Linux 2 的帮助，请参阅更新计算环境- AWS Batch。

启动模板支持

AWS Batch 支持在您的 EC2 计算环境中使用 Amazon EC2 启动模板。使用启动模板，您无需创建自定义 AMI 即可修改 AWS Batch 计算资源的默认配置。

> **Note**
>
> F AWS argate 资源不支持启动模板。

您必须先创建启动模板，然后才能将其与计算环境关联。您可以在 Amazon EC2 控制台创建启动模板。或者，您可以使用 AWS CLI 或 S AWS DK。例如，以下 JSON 文件表示一个启动模板，该模板可调整默认 AWS Batch 计算资源 AMI 的 Docker 数据量的大小，并将其设置为加密。

```
{
    "LaunchTemplateName": "increase-container-volume-encrypt",
    "LaunchTemplateData": {
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/xvda",
                "Ebs": {
                    "Encrypted": true,
                    "VolumeSize": 100,
                    "VolumeType": "gp2"
                }
            }
        ]
    }
}
```

您可以通过将 JSON 保存到调用的文件中`lt-data.json`并运行以下 AWS CLI 命令来创建之前的启动模板。

```
aws ec2 --region <region> create-launch-template --cli-input-json file://lt-data.json
```

有关启动模板的更多信息，请参阅 Amazon EC2 用户指南中的从启动模板启动实例。

如果使用启动模板来创建计算环境，则可以将以下现有计算环境参数移至启动模板：

> **Note**
>
> 假设在启动模板和计算环境配置中同时指定其中任意参数（Amazon EC2 标签除外）。然后，计算环境参数优先。Amazon EC2 标签在启动模板和计算环境配置之间合并。如果标签键发生冲突，则计算环境配置中的值优先。

- Amazon EC2 密钥对
- Amazon EC2 AMI ID
- 安全组 ID
- Amazon EC2 标签

以下启动模板参数将被忽略 AWS Batch：

- 实例类型（在创建计算环境时指定所需的实例类型）
- 实例角色（在创建计算环境时指定所需的实例角色）
- 网络接口子网（在创建计算环境时指定所需的子网）
- 实例市场选项（AWS Batch 必须控制竞价型实例配置）
- 禁用 API 终止（AWS Batch 必须控制实例生命周期）

AWS Batch 仅在基础架构更新期间使用新的启动模板版本更新启动模板。有关更多信息，请参阅 更新计算环境。

启动模板中的 Amazon EC2 用户数据

在启动实例时，您可以使用由cloud-init运行的启动模板中提供 Amazon EC2 用户数据。您的用户数据可以执行常见的配置方案，包括但不限于：

- 包含用户或组
- 安装程序包
- 创建分区和文件系统

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。这是因为您的用户数据与配置计算资源所需的其他 AWS Batch 用户数据合并。您可以将多个用户数据块合并到一个 MIME 分段文件中。例如，您可能希望将配置 Docker 进程守护程序的云 boothook 与为 Amazon ECS 容器代理写入配置信息的用户数据 Shell 脚本合并。

如果您正在使用 AWS CloudFormation，则该AWS::CloudFormation::Init类型可以与 cfn-in it 帮助脚本一起使用以执行常见的配置场景。

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="==BOUNDARY=="`

- 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：`--==BOUNDARY==`必须将此边界之前的行留空。
  - 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。必须将内容类型声明之后的行留空。
  - 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：`--==BOUNDARY==--`必须将此闭合边界之前的行留空。

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。

> **Note**
>
> 如果将用户数据添加到 Amazon EC2 控制台中的启动模板，则可以将其作为纯文本粘贴或从文件进行上载。或者，您可以从文件上传它。如果您使用 AWS CLI 或 AWS SDK，则必须先对用户数据进行base64编码，然后在调用 T CreateLaunchem plat `UserData` e 时将该字符串作为参数值提交，如此 JSON 文件所示。
>
> ```
> {
>     "LaunchTemplateName": "base64-user-data",
>     "LaunchTemplateData": {
>         "UserData":
>  "ewogICAgIkxhdW5jaFRlbXBsYXRlTmFtZSI6ICJpbmNyZWFzZS1jb250YWluZXItdm9sdW..."
>     }
> }
> ```

示例

- 示例：挂载现有 Amazon EFS 文件系统
- 示例：覆盖默认 Amazon ECS 容器代理配置
- 示例：挂载适用于 Lustre 的 Amazon FSx文件系统

示例：挂载现有 Amazon EFS 文件系统

Example

此示例 MIME 分段文件将配置计算资源以安装 amazon-efs-utils 程序包并在 /mnt/efs 处装载现有 Amazon EFS 文件系统。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- amazon-efs-utils

runcmd:
- file_system_id_01=fs-abcdef123
- efs_directory=/mnt/efs

- mkdir -p ${efs_directory}
- echo "${file_system_id_01}:/ ${efs_directory} efs tls,_netdev" >> /etc/fstab
- mount -a -t efs defaults

--==MYBOUNDARY==--
```

示例：覆盖默认 Amazon ECS 容器代理配置

Example

此示例 MIME 分段文件将覆盖计算资源的默认 Docker 映像清除设置。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo ECS_IMAGE_CLEANUP_INTERVAL=60m >> /etc/ecs/ecs.config
echo ECS_IMAGE_MINIMUM_CLEANUP_AGE=60m >> /etc/ecs/ecs.config

--==MYBOUNDARY==--
```

示例：挂载适用于 Lustre 的 Amazon FSx文件系统

Example

此示例 MIME 分段文件将配置计算资源，以从 Extras Library 安装 `lustre2.10` 程序包，并在 `/scratch` 处以 `fsx` 的装载名装载现有 FSx for Lustre 文件系统。此示例是 Amazon Linux 2 的示例。有关其他 Linux 发行版的安装说明，请参阅适用于 Lustre 的 Amazon FSx用户指南中的安装 Lustre 客户端。有关更多信息，请参阅适用于 Lustre 的 Amazon FSx用户指南中的自动挂载 Amazon FSx 文件系统。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- file_system_id_01=fs-0abcdef1234567890
- region=us-east-2
- fsx_directory=/scratch
- amazon-linux-extras install -y lustre2.10
- mkdir -p ${fsx_directory}
- mount -t lustre ${file_system_id_01}.fsx.${region}.amazonaws.com@tcp:fsx
 ${fsx_directory}

--==MYBOUNDARY==--
```

在容器属性的 volumes 和 mountPoints 成员中，装载点必须映射到容器中。

```
{
    "volumes": [
        {
            "host": {
                "sourcePath": "/scratch"
            },
            "name": "Scratch"
        }
    ],
    "mountPoints": [
        {
            "containerPath": "/scratch",
            "sourceVolume": "Scratch"
```

```
        }
    ],
}
```

创建计算环境

您需要先创建一个计算环境 AWS Batch，然后才能在中运行作业。您可以创建托管计算环境，在其中根据您的规格 AWS Batch 管理环境中的 Amazon EC2 实例或 AWS Fargate 资源。或者，您可以创建一个非托管计算环境，在其中处理环境中的 Amazon EC2 实例配置。

Important

在以下情况下，不支持 Fargate 竞价型实例：

- 在采用 ARM64 架构的 Amazon Linux 容器上。
- Windows containers on AWS Fargate

在这些情况下，如果将作业提交到仅使用 Fargate Spot 计算环境的作业队列，则作业队列将被阻止。

目录

使用 AWS Fargate 资源创建托管计算环境

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择计算环境。
4. 选择创建。
5. 配置计算环境。

> **Note**
>
> Windows containers on AWS Fargate 作业的计算环境必须至少有一个 vCPU。

a. 对于计算环境配置，请选择 Fargate。

b. 对于名称，为计算环境指定唯一名称。名称最长可以包含 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

c. 对于服务角色，请选择服务相关角色，该角色允许 AWS Batch 服务代表您调用所需 AWS 的 API 操作。例如，选择 AWSServiceRoleForBatch。有关更多信息，请参阅 的服务相关角色权限 AWS Batch。

d. （可选）展开标签。要添加标签，请选择 Add tag（添加标签）。然后，输入一个键和可选的值。选择 Add tag（添加标签）。

e. 选择下一页。

6. 在实例配置部分：

a. （可选）要使用 Fargate Spot 容量，请打开 Fargate Spot。有关 Fargate Spot 的信息，请参阅使用 Amazon EC2 Spot and Fargate_SPOT。

b. 对于 Maximum vCPUs，选择您的计算环境可以横向扩展到的 vCPU 的最大数目，而无论作业队列需求如何。

c. 选择下一页。

7. 配置网络。

> **Important**
>
> 计算资源需要访问才能与 Amazon ECS 服务端点通信。这可以通过接口 VPC 端点或具有公共 IP 地址的计算资源实现。
> 有关接口 VPC 端点的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 接口 VPC 端点(AWS PrivateLink)。
> 如果没有配置接口 VPC 端点，并且计算资源没有公共 IP 地址，则必须使用网络地址转换 (NAT) 来提供这种访问。有关更多信息，请参阅《Amazon VPC 用户指南》中的。???有关更多信息，请参阅 the section called “创建 VPC”。

a. 对于虚拟私有云（VPC）ID，请选择要在其中启动实例的 VPC。

b. 对于子网，选择要使用的子网。默认情况下，选定的 VPC 中的所有子网都可用。

> **Note**
>
> AWS Batch 在 Fargate 上目前不支持 Local Zones。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中 Local Zones、Wavelength Zones 和 AWS Outposts中的 Amazon ECS 集群。

c. 对于 Security groups，选择要附加到实例的安全组。默认情况下，将选择您的 VPC 的默认安全组。

d. 选择下一页。

8. 对于查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建计算环境。

若要使用 EC2 资源创建托管计算环境

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择计算环境。
4. 选择创建。
5. 配置环境。

   a. 在计算环境的配置中，选择 Amazon Elastic Compute Cloud (Amazon EC2)。

   b. 对于编排类型，请选择托管。

   c. 对于名称，为计算环境指定唯一名称。名称最长可以包含 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

   d. （可选）对于服务角色，请选择服务相关角色，该角色允许 AWS Batch 服务代表您调用所需 AWS 的 API 操作。例如，选择 AWSServiceRoleForBatch。有关更多信息，请参阅 的服务相关角色权限 AWS Batch。

   e. 对于 Instance role (实例角色)，请选择创建新的实例配置文件或使用附加了所需 IAM 权限的现有实例配置文件。此实例配置文件允许为您的计算环境创建的 Amazon ECS 容器实例代表您调用所需 AWS 的 API 操作。有关更多信息，请参阅 Amazon ECS 实例角色。如果您选择创建新实例配置文件，则将为您创建所需的角色 (`ecsInstanceRole`)。

   f. （可选）展开标签。

g. （可选）对于 EC2 标签，选择添加标签以向在计算环境中启动的资源添加标签。然后，输入一个键和可选的值。选择 Add tag（添加标签）。

h. （可选）在标签中，选择添加标签。然后，输入一个键和可选的值。选择 Add tag（添加标签）。

有关更多信息，请参阅 对AWS Batch资源加标签。

i. 选择下一页。

6. 在实例配置部分：

a. （可选）对于使用竞价型实例启用，请开启 Spot。有关更多信息，请参阅竞价型实例。

b. （仅限竞价型）对于按需价格最大百分比，请选择在启动实例之前与该实例类型的按需价格进行比较时竞价型实例价格可达到的最大百分比。例如，如果最高价为 20%，则竞价型价格必须低于该 EC2 实例的当前按需价格的 20%。您始终支付最低 (市场) 价格，并且绝不会高于您的最大百分比。如果将此字段留空，则默认值为按需价格的 100%。

c. （仅限 Spot）对于竞价型实例集角色，选择一个现有的 Amazon EC2 竞价型实例集 IAM 角色以应用于您的 Spot 计算环境。如果您没有现有的 Amazon EC2 竞价型实例集 IAM 角色，则必须先创建一个。有关更多信息，请参阅 Amazon EC2 竞价型实例集角色。

> **Important**
>
> 要在创建时标记您的竞价型实例，您的 Amazon EC2 竞价型队列 IAM 角色必须使用更新的 A mazonEC2 SpotFleet TaggingRole 托管策略。A mazonEC2 SpotFleet 角色托管策略没有标记竞价型实例所需的权限。有关更多信息，请参阅 创建时未标记的竞价型实例 和 the section called “对资源加标签”。

d. 对于 Minimum vCPUs，选择您的计算环境应保留的 vCPU 的最小数目，而无论作业队列需求如何。

e. 对于 Desired vCPUs，请选择您的计算环境在启动时应使用的 vCPU 数量。当作业队列需求增大时， AWS Batch 会增加计算环境中所需的 vCPU 数量并添加 EC2 实例 (最高可达最大 vCPU 数)。当需求减少时， AWS Batch 会减少计算环境中所需的 vCPU 数量并删除实例 (减少至最小 vCPU 数)。

f. 对于 Maximum vCPUs，选择您的计算环境可以横向扩展到的 vCPU 的最大数目，而无论作业队列需求如何。

g. 对于允许的实例类型，选择可启动的 Amazon EC2 实例类型。您可以指定实例系列以在这些系列中启动任何实例类型（例如，`c5`、`c5n` 或 `p3`），或者，您可以指定系列中的特定大小（例如 `c5.8xlarge`）。Metal 实例类型不在实例系列中。例如，`c5` 不包括 `c5.metal`。还

可以通过选择 optimal 来选择符合作业队列需求的实例类型（从 C4、M4 和 R4 实例系列中）。

> **Note**
>
> 在创建一个计算环境时，为该计算环境选择的实例类型必须共享同一架构。例如，您不能在同一个计算环境中混用 x86 和 ARM 实例。

> **Note**
>
> AWS Batch 将根据任务队列中所需的数量扩展 GPU。要使用 GPU 计划，计算环境必须包含 p2, p3, p4, p5, g3, g3s, g4 或 g5 系列的实例类型。

> **Note**
>
> 目前，optimal 使用 C4、M4 和 R4 实例系列中的实例类型。如果没有 AWS 区域这些实例系列的实例类型，则使用C5M5、和R5实例系列中的实例类型。

h. 展开其他配置。

i. （可选）对于置放群组，输入置放群组名称以对计算环境中的资源进行分组。

j. （可选）对于 EC2 密钥对，请在连接到实例时选择公钥和私有密钥对作为安全凭证。有关 Amazon EC2 密钥对的更多信息，请参阅 Amazon EC2 密钥对和 Linux 实例。

k. 对于分配策略，选择在从允许的实例类型列表中选择实例类型时要使用的分配策略。对于 EC2 按需计算环境，BEST_FIT_PROGRESSIVE 通常是更好的选择，而对于 EC2 Spot 计算环境，SPOT_CAPACITY_OPTIMIZED 和 SPOT_PRICE_CAPACITY_OPTIMIZED 则是更好的选择。有关更多信息，请参阅 the section called “分配策略”。

l. （可选）对于 EC2 配置，请选择映像类型和映像 ID 覆盖值以提供信息， AWS Batch 以便为计算环境中的实例选择 Amazon 系统映像 (AMI)。如果未为每种图像类型指定镜像 ID 替换，则 AWS Batch 选择最近经过亚马逊ECS优化的 AMI。如果未指定图像类型，则对于非 GPU、非 G AWS raviton 实例，默认为 Amazon Linux 2。

> **Important**
>
> 要使用自定义 AMI，请选择映像类型，然后在映像 ID 覆盖框中输入自定义 AMI ID。

Amazon Linux 2

所有 AWS 基于 Graviton 的实例系列（例如、C6gM6g、和T4g）均为默认值R6g，并且可用于所有非 GPU 实例类型。

Amazon Linux 2（GPU）

所有 GPU 实例系列的默认值（例如P4和G4），并且可用于所有非 AWS 基于 Graviton 的实例类型。

Amazon Linux

可用于非 GPU、非 G AWS raviton 实例系列。对 Amazon Linux AMI 的标准支持已结束。有关更多信息，请参阅 Amazon Linux AMI。

> **Note**
>
> 您为计算环境选择的 AMI 必须与您希望用于该计算环境的实例类型的架构匹配。例如，如果您的计算环境使用 A1 实例类型，则您选择的计算资源 AMI 必须支持 Arm 实例。Amazon ECS 同时提供经过 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 Arm 版本。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的经过 Amazon ECS 优化的 Amazon Linux 2 AMI。

m. （可选）对于启动模板，选择现有的 Amazon EC2 启动模板来配置您的计算资源。模板的默认版本会自动填充。有关更多信息，请参阅 启动模板支持。

> **Note**
>
> 在启动模板中，您可以指定自己创建的自定义 AMI。

n. （可选）对于 Launch template version (启动模板版本)，输入 `$Default`、`$Latest` 或要使用的特定版本号。

> **Important**
>
> 如果启动模板的版本参数为 `$Default` 或 `$Latest`，则会在基础设施更新期间评估指定启动模板的默认版本或最新版本。如果默认选择了不同的 AMI ID 或选择了最新

版本的启动模板，则将在更新中使用该 AMI ID。有关更多信息，请参阅 the section called “更新 AMI ID”。

o. 选择下一页。

7. 在网络配置部分：

> **Important**
>
> 计算资源需要访问才能与 Amazon ECS 服务端点通信。这可以通过接口 VPC 端点或具有公共 IP 地址的计算资源实现。
> 有关接口 VPC 端点的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 接口 VPC 端点(AWS PrivateLink)。
> 如果没有配置接口 VPC 端点，并且计算资源没有公共 IP 地址，则必须使用网络地址转换 (NAT) 来提供这种访问。有关更多信息，请参阅《Amazon VPC 用户指南》中的。???有关更多信息，请参阅 the section called “创建 VPC”。

a. 对于虚拟私有云（VPC）ID，请选择要在其中启动实例的 VPC。

b. 对于子网，选择要使用的子网。默认情况下，选定的 VPC 中的所有子网都可用。

> **Note**
>
> AWS Batch 在 Amazon EC2 上支持 Local Zones。有关更多信息，请参阅 Amazon EC2 用户指南中的本地区域，以及本地区域、波长区域和《亚马逊弹性容器服务开发人员指南》 AWS Outposts中的 Amazon ECS 集群。

c. （可选）对于安全组，选择要附加到实例的安全组。默认情况下，将选择您的 VPC 的默认安全组。

8. 选择下一页。

9. 对于查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建计算环境。

若要使用 EC2 资源创建非托管的计算环境

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。

2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在计算环境页面中，选择创建。
4. 配置环境。
   a. 在计算环境的配置中，选择 Amazon Elastic Compute Cloud (Amazon EC2)。
   b. 对于编排类型，请选择非托管。
5. 对于名称，为计算环境指定唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
6. （可选）对于服务角色，选择一个允许 AWS Batch 服务代表您调用所需的 AWS API 操作的角色。例如，选择 AWSBatchServiceRole。有关更多信息，请参阅the section called “使用服务相关角色”。
7. 对于 Maximum vCPUs，选择您的计算环境可以横向扩展到的 vCPU 的最大数目，而无论作业队列需求如何。
8. （可选）展开标签。要添加标签，请选择 Add tag（添加标签）。然后，输入一个键和可选的值。选择 Add tag（添加标签）。有关更多信息，请参阅 对AWS Batch资源加标签。
9. 选择下一页。
10. 对于查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建计算环境。

使用 Amazon EKS 资源创建托管计算环境

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择计算环境。
4. 选择创建。
5. 对于计算环境的配置，选择 Amazon Elastic Kubernetes Service (Amazon EKS)。
6. 对于名称，为计算环境指定唯一名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。
7. 对于 实例角色，请选择使用附加了所需 IAM 权限的现有实例配置文件。

> **Note**
>
> 要在 AWS Batch 控制台中创建计算环境，请选择具有`eks:ListClusters`和`eks:DescribeCluster`权限的实例配置文件。

8. 对于 EKS 集群，选择现有的 Amazon EKS 集群。
9. 在命名空间中，输入 Kubernetes 命名空间以对集群中的 AWS Batch 进程进行分组。
10. （可选）展开标签。选择添加标签，然后输入键值对。
11. 选择下一页。
12. （可选）对于使用 EC2 竞价型实例，请开启使用竞价型实例启用以使用 Amazon EC2 竞价型实例。
13. （仅限竞价型）对于按需价格最大百分比，请选择在启动实例之前与该实例类型的按需价格进行比较时竞价型实例价格可达到的最大百分比。例如，如果最高价为 20%，则竞价型价格必须低于该 EC2 实例的当前按需价格的 20%。您始终支付最低 (市场) 价格，并且绝不会高于您的最大百分比。如果将此字段留空，则默认值为按需价格的 100%。
14. （仅限竞价型）对于竞价型实例集角色，请为 SPOT 计算环境选择 Amazon EC2 竞价型实例集 IAM 角色。

> **Important**
>
> 如果将分配策略设置为 BEST_FIT 或者未指定时，需要使用该角色。

15. （可选）对于最小 vCPU 数，选择您的计算环境应保留的 vCPU 的最小数目，而无论作业队列需求如何。
16. （可选）对于最大 vCPU 数，选择您的计算环境可以横向扩展到的 vCPU 的最大数目，而无论作业队列需求如何。
17. 对于允许的实例类型，选择可启动的 Amazon EC2 实例类型。您可以指定实例系列以在这些系列中启动任何实例类型（例如，c5、c5n 或 p3），或者，您可以指定系列中的特定大小（例如 `c5.8xlarge`）。Metal 实例类型不在实例系列中。例如，c5 不包括 `c5.metal`。还可以通过选择 `optimal` 来选择符合作业队列需求的实例类型（从 C4、M4 和 R4 实例系列中进行选择）。

> **Note**
>
> 在创建一个计算环境时，为该计算环境选择的实例类型必须共享同一架构。例如，您不能在同一个计算环境中混用 x86 和 ARM 实例。

> **Note**
>
> AWS Batch 根据任务队列中所需的数量扩展 GPU。要使用 GPU 计划，计算环境必须包含 p2, p3, p4, p5, g3, g3s, g4 或 g5 系列的实例类型。

> **Note**
>
> 目前，`optimal` 使用 C4、M4 和 R4 实例系列中的实例类型。如果没有 AWS 区域 这些实例系列的实例类型，则使用C5M5、和R5实例系列中的实例类型。

18. （可选）展开其他配置。

    a. （可选）对于置放群组，输入置放群组名称以对计算环境中的资源进行分组。

    b. 对于分配策略，选择 BEST_FIT_PROGRESSIVE。

    c. （可选）对于亚马逊机器映像（AMI）配置，请选择添加亚马逊机器映像（AMI）配置。然后，选择映像类型，输入映像 ID 覆盖和 Kubernetes 版本。

    > **Important**
    >
    > 要使用自定义 AMI，请选择映像类型，然后在映像 ID 覆盖框中输入自定义 AMI ID。

    > **Note**
    >
    > 如果未为每种图像类型指定镜像 ID 替换，则 AWS Batch 选择最近经过亚马逊ECS优化的 AMI。如果未指定图像类型，则对于非 GPU、非 G AWS raviton 实例，默认为 Amazon Linux 2。

Amazon Linux 2

所有 AWS 基于 Graviton 的实例系列（例如、C6gM6g、和T4g）均为默认值R6g，并且可用于所有非 GPU 实例类型。

Amazon Linux 2（GPU）

所有 GPU 实例系列（例如P4和G4）均为默认值，可用于所有非 AWS 基于 Graviton 的实例类型。

d. （可选）对于启动模板，请选择现有启动模板。

e. （可选）对于启动模板版本，输入 **$Default**、**$Latest** 或版本号码。

19. 选择下一页。
20. 对于虚拟私有云（VPC）ID，选择要启动实例的 VPC。
21. 对于子网，选择要使用的子网。默认情况下，选定的 VPC 中的所有子网都可用。

> **Note**
>
> AWS Batch 在亚马逊上，EKS 支持 Local Zones。有关更多信息，请参阅《亚马逊 EKS 用户指南》中的 Amazon EKS 和 L AWS ocal Zones。

22. （可选）对于安全组，选择要附加到实例的安全组。默认情况下，将选择您的 VPC 的默认安全组。
23. 选择下一页。
24. 对于查看，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择创建计算环境。

计算环境模板

以下示例显示了空的计算环境模板。可以使用此模板创建计算环境，随后可将计算环境保存到文件并与 AWS CLI `--cli-input-json`选项结合使用。有关这些参数的更多信息，请参阅AWS Batch API 参考中的CreateComputeEnvironment。

```
{
    "computeEnvironmentName": "",
    "type": "UNMANAGED",
    "state": "DISABLED",
```

```
    "unmanagedvCpus": 0,
    "computeResources": {
        "type": "EC2",
        "allocationStrategy": "BEST_FIT_PROGRESSIVE",
        "minvCpus": 0,
        "maxvCpus": 0,
        "desiredvCpus": 0,
        "instanceTypes": [
            ""
        ],
        "imageId": "",
        "subnets": [
            ""
        ],
        "securityGroupIds": [
            ""
        ],
        "ec2KeyPair": "",
        "instanceRole": "",
        "tags": {
            "KeyName": ""
        },
        "placementGroup": "",
        "bidPercentage": 0,
        "spotIamFleetRole": "",
        "launchTemplate": {
            "launchTemplateId": "",
            "launchTemplateName": "",
            "version": ""
        },
        "ec2Configuration": [
            {
                "imageType": "",
                "imageIdOverride": "",
                "imageKubernetesVersion": ""
            }
        ]
    },
    "serviceRole": "",
    "tags": {
        "KeyName": ""
    },
    "eksConfiguration": {
        "eksClusterArn": "",
```

```
        "kubernetesNamespace": ""
    }
}
```

> **Note**
>
> 可以使用以下AWS CLI命令生成上述计算环境模板。
>
> ```
> $ aws batch create-compute-environment --generate-cli-skeleton
> ```

计算环境参数

计算环境分为几个基本组成部分：计算环境的名称、类型和状态、计算资源定义（如果是托管计算环境）、Amazon EKS 配置（如果它使用 Amazon EKS 资源）、用于向其提供 IAM 权限的服务角色以及计算环境的标签。 AWS Batch

主题

计算环境名称

`computeEnvironmentName`

您的计算环境的名称。名称长度不超过 128 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

类型：字符串

必需：是

类型

`type`

计算环境的类型。选择MANAGED让 AWS Batch 管理您定义的 EC2 或 Fargate 计算资源。有关更多信息，请参阅 计算资源。选择 UNMANAGED 可管理您自己的 EC2 计算资源。

类型：字符串

有效值：MANAGED | UNMANAGED

必需：是

状态

`state`

计算环境的状态。

如果状态为ENABLED，则 AWS Batch 调度器会尝试在环境中放置作业。这些作业来自计算资源上的关联作业队列。如果计算环境是托管计横向扩展算环境，则实例会根据作业队列需求自动横向扩展或向内扩展。

如果状态为DISABLED，则 AWS Batch 调度器不会尝试在环境中放置作业。处于 STARTING 或 RUNNING 状态的作业将继续正常运行。处于 DISABLED 状态的托管计算环境无法横向扩展。

> **Note**
>
> 处于 DISABLED 状态的计算环境可能会继续产生账单费用。为防止额外收费，请关闭计算环境，然后将其删除。有关更多信息，请参阅 DeleteComputeEnvironment AWS Batch API 参考和《AWS Billing 用户指南》中的 “避免意外费用”。

当实例处于空闲状态时，该实例会缩小到 minvCpus 值。但是，实例大小不会变化。例如，假设一个 minvCpus 值为 4 且 desiredvCpus 值为 36 的 c5.8xlarge 实例。此实例不会缩减为 c5.large 实例。

类型：字符串

有效值：ENABLED | DISABLED

必需：否

计算资源

`computeResources`

由计算环境托管的计算资源的详细信息。有关更多信息，请参阅 计算环境。

类型：ComputeResource 对象

必需：此参数是托管计算环境所必需的

`type`

计算环境的类型。您可以选择使用 EC2 按需实例 (EC2) 和 EC2 竞价型实例(SPOT)，或者在托管计算环境中使用 Fargate 容量 (FARGATE) 和 Fargate 竞价型容量 (FARGATE_SPOT)。如果选择 SPOT，则还必须使用 `spotIamFleetRole` 参数指定 Amazon EC2 竞价型实例集角色。有关更多信息，请参阅 Amazon EC2 竞价型实例集角色。

有效值：`EC2` | `SPOT` | `FARGATE` | `FARGATE_SPOT`

必需：是

`allocationStrategy`

如果无法分配最佳匹配 EC2 实例类型的足够实例时，要用于计算资源的分配策略。这可能是由于实例类型在 AWS 区域 或 Amazon EC2 服务限制中的可用性所致。有关更多信息，请参阅 分配策略。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

`BEST_FIT`（默认值）

AWS Batch 选择最适合任务需求的实例类型，优先选择成本最低的实例类型。如果所选实例类型的其他实例不可用，则 AWS Batch 等待其他实例可用。如果没有足够可用的实例，或如果您达到 Amazon EC2 服务限制，则其他作业将在当前运行的作业完成以后不会运行。此分配策略可降低成本，但会限制扩展。如果将竞价型实例集与 `BEST_FIT` 一起使用，则必须

指定竞价型实例集 IAM 角色。使用 BEST_FIT 分配策略的计算资源不支持基础架构更新，也无法更新某些参数。有关更多信息，请参阅 更新计算环境。

> **Note**
>
> 使用 Amazon EKS 资源的计算环境不支持 BEST_FIT。

BEST_FIT_PROGRESSIVE

使用足够大的、能够满足队列中作业要求的额外实例类型。首选每单位 vCPU 成本较低的实例类型。如果之前选择的实例类型的其他实例不可用，请 AWS Batch 选择新的实例类型。

SPOT_CAPACITY_OPTIMIZED

（仅适用竞价型实例计算资源）使用足够大的其他实例类型，以满足队列中作业的要求。首选不太可能被中断的实例类型。

SPOT_PRICE_CAPACITY_OPTIMIZED

（仅适用于竞价型实例计算资源）价格和容量优化分配策略同时考虑价格和容量，以选择中断可能性最小、价格尽可能低的竞价型实例池。

> **Note**
>
> 建议在大多数情况下使用SPOT_PRICE_CAPACITY_OPTIMIZED而不是SPOT_CAPACITY_OPTIMIZED。

使用BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED、和SPOT_PRICE_CAPACITY_OPTIMIZED使用按需实例或竞价型实例的BEST_FIT策略以及使用竞价型实例的策略， AWS Batch 可能需要超过，maxvCpus才能满足您的容量需求。在这种情况下， AWS Batch 永远不要超过maxvCpus一个实例。

有效值：BEST_FIT | BEST_FIT_PROGRESSIVE | SPOT_CAPACITY_OPTIMIZED | SPOT_PRICE_CAPACITY_OPTIMIZED

必需：否

minvCpus

环境保留的 vCPU 的最小数量（即使计算环境为 DISABLED）。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：整数

必需：否

`maxvCpus`

AWS Batch 计算环境可以支持的最大 vCPU 数量。

> **Note**
>
> 使用`BEST_FIT_PROGRESSIVE`、`SPOT_CAPACITY_OPTIMIZED`、和使用按需实例或竞价型实例的`SPOT_PRICE_CAPACITY_OPTIMIZED`分配`BEST_FIT`策略以及使用竞价型实例的策略 AWS Batch 可能需要超过，`maxvCpus`才能满足您的容量需求。在这种情况下， AWS Batch 永远不要超过`maxvCpus`一个实例。例如，只 AWS Batch 使用计算环境中指定的实例中的一个实例。

类型：整数

必需：否

`desiredvCpus`

计算环境中所需的 vCPU 数量。 AWS Batch 根据作业队列需求在最小值和最大值之间修改此值。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：整数

必需：否

`instanceTypes`

可启动的实例类型。此参数不适用于在 Fargate 资源上运行的作业。不要指定此参数。您可以指定实例系列以在这些系列中启动任何实例类型（例如，`c5`、`c5n` 或 `p3`），或者，您可以指定系列中的特定大小（例如 `c5.8xlarge`）。请注意，裸机实例类型不包括在实例系列中（例如，`c5` 不包括 `c5.metal`）。还可以通过选择 `optimal` 来选择符合作业队列需求的实例类型（从 C4、M4 和 R4 实例系列中）。

> **Note**
>
> 在创建一个计算环境时，为该计算环境选择的实例类型必须共享同一架构。例如，您不能在同一个计算环境中混用 x86 和 ARM 实例。

> **Note**
>
> 目前，`optimal` 使用 C4、M4 和 R4 实例系列中的实例类型。在没有属于这些实例系列的实例类型的 AWS 区域 中，使用 C5、M5 和 R5 实例系列的实例类型。

类型：字符串数组

必需：是

`imageId`

此参数已被弃用。

用于计算环境中启动的实例的 Amazon 系统映像 (AMI) ID。此参数被 `Ec2Configuration` 结构的 `imageIdOverride` 成员覆盖。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

> **Note**
>
> 您为计算环境选择的 AMI 必须与您希望用于该计算环境的实例类型的架构匹配。例如，如果您的计算环境使用 A1 实例类型，则您选择的计算资源 AMI 必须支持 Arm 实

例。Amazon ECS 同时提供经过 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 Arm 版本。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的经过 Amazon ECS 优化的 Amazon Linux 2 AMI。

类型：字符串

必需：否

`subnets`

计算资源在其中启动的 VPC 子网。这些子网必须位于同一 VPC 中。Fargate 计算资源最多可以包含 16 个子网。有关更多信息，请参阅 Amazon VPC 用户指南 中的 VPC 和子网。

> **Note**
>
> AWS Batch 在亚马逊 EC2 和亚马逊 AWS Batch 上，EKS 都支持 Local Zones。有关更多信息，请参阅 Amazon EC2 用户指南中的本地区域、亚马逊 EKS 用户指南中的 Amazon EKS 和 AWS 本地区域，以及本地区域、波长区域和《亚马逊弹性容器服务开发者指南》 AWS Outposts中的 Amazon ECS 集群。
> AWS Batch 在 Fargate 上目前不支持 Local Zones。

更新计算环境时，如果您提供的 VPC 子网列表为空，则 Fargate 和 EC2 计算资源的结果行为会有所不同。对于 Fargate 计算资源，如果提供空列表，则会像未指定此参数一样处理，并且不进行任何更改。对于 EC2 计算资源，提供空列表会将 VPC 子网从计算资源中删除。如果您更改 VPC 子网，则需要更新计算环境的基础设施。这同样适用于 Fargate 和 EC2 计算资源。有关更多信息，请参阅 更新计算环境。

类型：字符串数组

必需：是

`securityGroupIds`

与计算环境中启动的实例关联的 Amazon EC2 安全组。必须在 `securityGroupIds` 中或使用 `launchTemplate` 中引用的启动模板指定一个或多个安全组。对于在 Fargate 资源上运行的作业，此参数是必需的，且必须至少包含一个安全组。（Fargate 不支持启动模板。）如果同时使用 `securityGroupIds` 和 `launchTemplate` 指定安全组，将使用 `securityGroupIds` 中的值。

在更新计算环境时，如果提供一个空的安全组列表，则 Fargate 和 EC2 计算资源之间的行为会有所不同。对于 Fargate 计算资源，如果提供空列表，则会像未指定此参数一样处理，并且不进行任何更改。对于 EC2 计算资源，提供空列表会将安全组从计算资源中删除。如果您更改安全组，则需要更新计算环境的基础设施。这同样适用于 Fargate 和 EC2 计算资源。有关更多信息，请参阅 更新计算环境。

类型：字符串数组

必需：是

`ec2KeyPair`

用于计算环境中启动的实例的 EC2 密钥对。您可以使用此密钥对通过 SSH 登录您的实例。更新计算环境时，更改 EC2 密钥对需要更新计算环境的基础设施。有关更多信息，请参阅 更新计算环境。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：字符串

必需：否

`instanceRole`

要附加到计算环境中的 Amazon EC2 实例的 Amazon ECS 实例配置文件。此参数不适用于在 Fargate 资源上运行的作业。不要指定此参数。您可以为实例配置文件指定短名称或完整的 Amazon 资源名称 (ARN)。例如，`ecsInstanceRole` 或 `arn:aws:iam::`*`aws_account_id`*`:instance-profile/ecsInstanceRole`。有关更多信息，请参阅 Amazon ECS 实例角色。

更新计算环境时，更改此设置需要更新计算环境的基础设施。有关更多信息，请参阅 更新计算环境。

类型：字符串

必需：否

`tags`

要应用于计算环境中启动的 EC2 实例的键/值对标签。例如，您可以指定 `"Name": "AWS Batch Instance - C4OnDemand"` 作为标签，以便计算环境中的每个实例均具有

此名称。这有助于在 Amazon EC2 控制台中识别您的 AWS Batch 实例。使用 AWS BatchListTagsForResourceAPI 操作时看不到这些标签。

更新计算环境时，更改 EC2 标签需要更新计算环境的基础设施。有关更多信息，请参阅 更新计算环境。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：字符串到字符串映射

必需：否

`placementGroup`

要与计算资源关联的 Amazon EC2 置放群组。此参数不适用于在 Fargate 资源上运行的作业。不要指定此参数。如果打算将多节点并行作业提交到计算环境，请考虑创建一个集群置放群组，并将其与计算资源相关联。这会保证实例逻辑分组上的多节点并行作业位于单个可用区中，同时提供较高的网络流量潜力。有关更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的放置组。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：字符串

必需：否

`bidPercentage`

在启动实例之前，与该实例类型的按需价格进行比较时 EC2 竞价型实例价格可以达到的最大百分比。例如，如果最大百分比为 20%，则竞价型价格必须低于该 EC2 实例的当前按需价格的 20%。您始终支付最低 (市场) 价格，并且绝不会高于您的最大百分比。如果将此字段留空，则默认值为按需价格的 100%。对于大多数使用案例，我们建议将此字段留空。

更新计算环境时，更改竞价百分比需要更新计算环境的基础设施。有关更多信息，请参阅 更新计算环境。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

必需：否

`spotIamFleetRole`

应用于 SPOT 计算环境的 Amazon EC2 竞价型实例集 IAM 角色的 Amazon 资源名称 (ARN)。如果将分配策略设置为 BEST_FIT，或者未指定分配策略，则需要使用该角色。有关更多信息，请参阅 Amazon EC2 竞价型实例集角色。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

> **Important**
>
> 要在创建时标记您的竞价型实例，此处指定的竞价型队列 IAM 角色必须使用更新的 AmazonEC2 SpotFleet TaggingRole 托管策略。之前推荐的 AmazonEC2 SpotFleet 角色托管策略没有标记竞价型实例所需的权限。有关更多信息，请参阅 创建时未标记的竞价型实例。

类型：字符串

必需：此参数对于 SPOT 计算环境是必需的。

`launchTemplate`

要与计算资源关联的可选启动模板。此参数不适用于在 Fargate 资源上运行的作业。不要指定此参数。在 CreateComputeEnvironment 或 UpdateComputeEnvironment API 操作中指定的任何其他计算资源参数将覆盖启动模板中的相同参数。要使用启动模板，您必须在请求中指定启动模板 ID 或启动模板名称，但不能同时指定两者。有关更多信息，请参阅 启动模板支持。

更新计算环境时，要移除自定义启动模板并使用默认启动模板，请将启动模板规范的 `launchTemplateId` 或 `launchTemplateName` 成员设置为空字符串。从计算环境中移除启动模板时，不会删除启动模板中指定的 AMI（如果它是所使用的 AMI 的时候）。要更新从启动

模板中选择的 AMI，必须将 updateToLatestImageVersion 参数设置为 true。更新计算环境时，更改启动模板需要更新计算环境的基础设施。有关更多信息，请参阅 更新计算环境。

类型：LaunchTemplateSpecification

object

必需：否

launchTemplateId

启动模板的 ID。

类型：字符串

必需：否

launchTemplateName

启动模板的名称。

类型：字符串

必需：否

version

启动模板的版本号，$Latest 或 $Default。

如果值为 $Latest，则使用启动模板的最新版本。如果值为 $Default，则使用启动模板的默认版本。在基础架构更新期间，如果为计算环境指定了$Latest或$Default，则会 AWS Batch 重新评估启动模板版本，并可能使用其他版本的启动模板。即使更新中未指定启动模板，也会出现这种情况。

默认值：$Default。

类型：字符串

必需：否

ec2Configuration

提供用于在 EC2 计算环境中为实例选择 Amazon 系统映像 (AMI) 的信息。如果未指定 Ec2Configuration，则默认值为 Amazon Linux 2 (ECS_AL2)。在 2021 年 3 月 31 日之前，非 GPU、非 G AWS raviton 实例的默认设置为 Amazon Linux (ECS_AL1)。

更新计算环境时，更改此参数需要更新计算环境的基础设施。有关更多信息，请参阅 更新计算环境。

> **Note**
>
> 此参数不适用于在 Fargate 资源上运行的作业。

类型：Ec2Configuration 对象数组

必需：否

`imageIdOverride`

用于在计算环境中启动的与映像类型匹配的实例的 AMI ID。此设置会覆盖 `computeResource` 对象中的 `imageId` 集。

类型：字符串

必需：否

`imageKubernetesVersion`

计算环境的 Kubernetes 版本。如果您未指定值，则将使用 AWS Batch 支持的最新版本。

类型：字符串

长度限制：最小长度为 1。最大长度为 256。

必需：否

`imageType`

要与实例类型匹配以选择 AMI 的映像类型。`ECS` 和 `EKS` 资源的受支持值不同。

ECS

如果未指定 `imageIdOverride` 参数，则使用最近的经过 Amazon ECS 优化的 Amazon Linux 2 AMI（`ECS_AL2`）。如果在更新中指定了新的图像类型，但既没有指定`imageId`也没有指定`imageIdOverride`参数，则将使用支持的最新 Amazon ECS 针对该图像类型进行优化 AWS Batch 的 AMI。

ECS_AL2

Amazon Linux 2：所有非 GPU 实例系列的默认值。

ECS_AL2_NVIDIA

Amazon Linux 2 (GPU)：所有 GPU 实例系列的默认值（例如P4和G4），并且可用于所有非 AWS 基于 Graviton 的实例类型。

ECS_AL1

Amazon Linux。亚马逊 Linux 已达到标准支持。 end-of-life 有关更多信息，请参阅 Amazon Linux AMI。

EKS

如果未指定 imageIdOverride 参数，则使用最近的 Amazon EKS 优化版 Amazon Linux AMI (EKS_AL2)。如果在更新中指定了新的图像类型，但既未指定imageId也未指定imageIdOverride参数，则将使用针对该图像类型 AWS Batch 支持的最新 Amazon EKS 优化的 AMI。

EKS_AL2

Amazon Linux 2：所有非 GPU 实例系列的默认值。

EKS_AL2_NVIDIA

Amazon Linux 2（加速）：所有 GPU 实例系列（例如P4和G4）的默认设置，可用于所有非 AWS 基于 Graviton 的实例类型。

类型：字符串

长度限制：最小长度为 1。最大长度为 256。

必需：是

Amazon EKS 配置

支持 AWS Batch 计算环境的 Amazon EKS 集群的配置。集群必须先存在，才能创建计算环境。

`eksClusterArn`

Amazon EKS 集群的 Amazon 资源名称（ARN）。例如，`arn:aws:eks:`*`us-east-1`*`:`*`123456789012`*`:cluster/`*`ClusterForBatch`*。

类型：字符串

必需：是

`kubernetesNamespace`

Amazon EKS 集群的命名空间。 AWS Batch 管理此命名空间中的 pod。值不能为空或为 null。长度必须少于 64 个字符，不能设置为 `default`，不能以“kube-”开头，并且必须匹配此正则表达式：`^[a-z0-9]([-a-z0-9]*[a-z0-9])?$`。有关更多信息，请参阅 Kubernetes 文档中的命名空间。

类型：字符串

必需：是

类型：EksConfiguration对象

必需：否

服务角色

`serviceRole`

允许 AWS Batch 代表您调用其他 AWS 服务的 IAM 角色的完整亚马逊资源名称 (ARN)。有关更多信息，请参阅 将服务相关角色用于 AWS Batch。我们建议您不指定服务角色。这样，就 AWS Batch 使用AWSServiceRoleForBatch服务相关角色。

> **Important**
>
> 如果您的账户已经创建了 AWS Batch 服务相关角色 (AWSServiceRoleForBatch)，则除非您在此处指定角色，否则默认情况下，该角色将用于您的计算环境。如果您的账户中不存在 AWS Batch 服务相关角色，且此处未指定任何角色，则该服务会尝试在您的账户中创建 AWS Batch 服务相关角色。有关 AWSServiceRoleForBatch 服务相关角色的更多信息，请参阅的服务相关角色权限 AWS Batch。
>
> 如果使用 AWSServiceRoleForBatch 服务相关角色创建计算环境，则不能将其更改为使用常规 IAM 角色。同样，如果使用常规 IAM 角色创建计算环境，则不能将其更改为使用 AWSServiceRoleForBatch 服务相关角色。要更新需要更新基础架构才能更改的计算环境的参数，必须使用 AWSServiceRoleForBatch 服务相关角色。有关更多信息，请参阅 更新计算环境。

如果您的指定角色的路径并非 `/`，请确保指定完整角色 ARN（推荐）或将此路径作为角色名称的前缀。

> **Note**
>
> 根据您创建 AWS Batch 服务角色的方式，其 Amazon 资源名称 (ARN) 可能包含`service-role`路径前缀。当您仅指定服务角色的名称时， AWS Batch 假设您的 ARN 不使用`service-role`路径前缀。因此，我们建议您在创建计算环境时指定服务角色的完整 ARN。

类型：字符串

必需：否

Tags

`tags`

要与计算环境关联的键值对标签。有关更多信息，请参阅 对AWS Batch资源加标签。

类型：字符串到字符串映射

必需：否

EC2 配置

AWS Batch 使用适用于 EC2 和 EC2 竞价计算环境的 Amazon ECS 优化的 AMI。默认为 Amazon Linux 2 (`ECS_AL2`)。在 2021 年 3 月 31 日之前，非 GPU、非 G AWS raviton 实例的默认设置为 Amazon Linux (`ECS_AL1`)。

> **Note**
>
> AWS Batch 还支持亚马逊 Linux 2023。

亚马逊 Linux AMI（也称为亚马逊 Linux 1）于 2023 年 12 月 31 日停产。 AWS Batch 已终止对亚马逊 Linux AMI 的支持，因为它从 2024 年 1 月 1 日起将不会收到任何安全更新或错误修复。有关 Amazon Linux 的更多信息 end-of-life，请参阅 A L 常见问题解答。

我们建议您将基于 Amazon Linux 的现有计算环境更新到 Amazon Linux 2023，以防止不可预见的工作负载中断，并继续接收安全和其他更新。

您使用亚马逊 Linux AMI 的计算环境可能会在 2023 年 end-of-life 12 月 31 日之后继续运行。但是，这些计算环境将不再收到来自的任何新软件更新、安全补丁或错误修复 AWS。之后，您有责任在 Amazon Linux AMI 上维护这些计算环境 end-of-life。我们建议将 AWS Batch 计算环境迁移到亚马逊 Linux 2023 或亚马逊 Linux 2，以保持最佳性能和安全性。

要获得 AWS Batch 从亚马逊 Linux AMI 迁移到亚马逊 Linux 2023 或亚马逊 Linux 2 的帮助，请参阅更新计算环境- AWS Batch

分配策略

创建托管计算环境时，从`instanceTypes`指定的实例类型 AWS Batch 中选择最适合任务需求的实例类型。分配策略定义了 AWS Batch 需要额外容量时的行为。此参数不适用于在 Fargate 资源上运行的作业。请勿指定此参数。

`BEST_FIT`（默认值）

AWS Batch 选择最适合任务需求的实例类型，优先选择成本最低的实例类型。如果所选实例类型的其他实例不可用，则 AWS Batch 等待其他实例可用。如果没有足够可用的实例，或如果用户达到 Amazon EC2 服务限额，则其他作业只有在当前正在运行的作业完成之后才会运行。此分配策略可降低成本，但会限制扩展。如果将竞价型实例集与`BEST_FIT`一起使用，则必须指定竞价型实例集 IAM 角色。更新计算环境时不支持`BEST_FIT`。有关更多信息，请参阅 更新计算环境。

> **Note**
>
> AWS Batch 管理您账户中的 AWS 资源。默认情况下，采用 BEST_FIT 分配策略的计算环境最初使用启动配置。但是，随着时间的推移，对新 AWS 账户使用启动配置将受到限制。因此，从 2024 年 4 月下旬开始，新创建的 BEST_FIT 计算环境将默认启动模板。如果您的服务角色缺乏管理启动模板的权限，则 AWS Batch 可以继续使用启动配置。现有计算环境将继续使用启动配置。

`BEST_FIT_PROGRESSIVE`

AWS Batch 选择足够大以满足队列中任务要求的其他实例类型。优先选择每个 vCPU 成本较低的实例类型。如果以前选择的实例类型没有可用的额外实例， AWS Batch 将选择新的实例类型。

`SPOT_CAPACITY_OPTIMIZED`

AWS Batch 选择一个或多个足以满足队列中任务要求的实例类型。优先选择不太可能被中断的实例类型。此分配策略仅适用于竞价型实例计算资源。

SPOT_PRICE_CAPACITY_OPTIMIZED

价格和容量优化分配策略同时考虑价格和容量，以选择中断可能性最小、价格尽可能低的竞价型实例池。此分配策略仅适用于竞价型实例计算资源。

> **Note**
>
> 建议在大多数情况下使用SPOT_PRICE_CAPACITY_OPTIMIZED而不是SPOT_CAPACITY_OPTIMIZED。

BEST_FIT_PROGRESSIVE和BEST_FIT策略使用按需实例或竞价型实例，SPOT_CAPACITY_OPTIMIZED和SPOT_PRICE_CAPACITY_OPTIMIZED策略使用竞价型实例。但是， AWS Batch 可能需要超出容量maxvCpus才能满足您的容量需求。在这种情况下， AWS Batch 永远不要超过maxvCpus一个实例。

更新计算环境

创建使用 EC2 资源的计算环境后，可以直接更新计算环境的许多设置。但是，更改某些设置需要AWS Batch替换计算环境中的实例。

对于使用 Fargate 资源的计算环境，可以更新以下内容。

- securityGroupIds
- subnets
- desiredvCpus
- maxvCpus
- minvCpus

AWS Batch有两种更新机制。第一种是扩展更新，即从计算环境中添加或删除实例。第二种是基础架构更新，即替换计算环境中的实例。基础架构更新比扩展更新所需的时间要长得多。

如果使用AWS Batch更新计算环境，则仅更改以下设置会导致扩展更新：所需的 vCPU (desiredvCpus)、最大 vCPU (maxvCpus)、最小 vCPU (minvCpus)、服务角色 (serviceRole) 和状态 (state)。

> **Note**
>
> 更新desiredvCpus设置时，该值必须介于minvCpus和maxvCpus值之间。
> 此外，更新的desiredvCpus值必须大于或等于当前的desiredvCpus值。有关更多信息，请参阅the section called “更新desiredvCpus设置时出现错误消息”。

如果在 UpdateComputeEnvironmentAPI 操作中更改了以下任何设置，则AWS Batch会启动基础架构更新。基础架构更新要求将服务角色设置为 AWSServiceRoleForBatch（默认），并且分配策略为BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED或SPOT_PRICE_CAPACITY_OPTIMIZED。不支持BEST_FIT。除了服务角色之外，所有可在扩展更新中更改的设置也可在基础架构更新中更改。

> **Note**
>
> 建议在大多数情况下使用SPOT_PRICE_CAPACITY_OPTIMIZED而不是SPOT_CAPACITY_OPTIMIZEDn。

在基础架构更新期间，计算环境的状态更改为UPDATING。使用更新的设置启动新实例。新作业被安排到新实例上。当前正在运行的作业将根据基础架构更新策略进行分配。有关更多信息，请参阅 AWS BatchAPI 参考中的UpdateComputeEnvironment和UpdatePolicy。

在UpdatePolicy数据类型中，请考虑以下情景：

> **Note**
>
> 在这些情景中，以下各项适用。实例终止时，正在运行的作业也会停止。默认情况下，不会重试这些任务。要在实例终止后重试其中一个作业，请配置任务重试策略。有关更多信息，请参阅AWS Batch《用户指南》中的the section called “自动作业重试”。

- 如果将terminateJobsOnUpdate设置设为true，则在基础架构更新期间将终止正在运行的作业。jobExecutionTimeoutMinutes设置将忽略。
- 如果将terminateJobsOnUpdate设置设为false，则在基础架构更新后，作业可以运行更多时间。该额外时间在jobExecutionTimeoutMinutes设置中配置。默认情况下，jobExecutionTimeoutMinutes设置为 30 分钟。

当计算环境中有可用容量时，将使用更新的设置启动新实例，并在新实例上启动作业。在使用旧设置的实例上完成所有作业后，旧实例就会终止。容量变为可用意味着所需的 vCPU 数量比最大 vCPU 数量少，至少少于最小实例类型所需的 vCPU 数量。

基础架构更新

要更改计算环境的某些设置，需要更新基础架构。如果更改了以下任何设置，则会启动基础架构更新：

> **Important**
>
> 计算环境必须使用 AWSServiceRoleForBatch 服务相关角色来进行需要更新基础架构的更改。如果计算环境使用服务相关角色，则无法将其更改为使用常规 IAM 角色。同样，如果计算环境具有常规 IAM 角色，则无法将其更改为使用服务相关角色。因此，只能在使用服务相关角色创建的计算环境上执行基础架构更新。

- 分配策略 (`allocationStrategy`，必须是`BEST_FIT_PROGRESSIVE`、`SPOT_CAPACITY_OPTIMIZED`或`SPOT_PRICE_CAPACITY_OPTIMIZED`。如果最初的分配策略是`BEST_FIT`，则不支持基础架构更新。)

  > **Note**
  >
  > 建议在大多数情况下使用`SPOT_PRICE_CAPACITY_OPTIMIZED`而不是`SPOT_CAPACITY_OPTIMIZEDn`。

- 出价百分比 (`bidPercentage`)
- EC2 配置 (`ec2Configuration`)
- 密钥对 (`ec2KeyPair`)
- 映像 ID (`imageId`)
- 实例角色 (`instanceRole`)
- 实例类型 (`instanceTypes`)
- 启动模板 (`launchTemplate`)
- 置放群组 (`placementGroup`)
- 安全组 (`securityGroupIds`)
- VPC 子网 (`subnets`)
- EC2 标签 (`tags`)

- 计算环境类型（type，可以是EC2或SPOT之一）
- 是否在基础架构更新期间更新到AWS Batch支持的最新 AMIupdateToLatestImageVersion

更新 AMI ID

在基础架构更新期间，计算环境的 AMI ID 可能会发生变化，这取决于是否在上述三种设置中的任何一种中指定了 AMI。AMI
在imageId（在computeResources）、imageIdOverride（在ec2Configuration）中指定，或者在launchTemplate指定的启动模板中指定。假设这些设置中都没有指定 AMI ID，且updateToLatestImageVersion设置是true。然后，AWS Batch支持的最新 Amazon ECS 优化 AMI 将用于任何基础架构更新。

如果至少在其中一个设置中指定了 AMI ID，则更新将取决于提供了更新前使用的 AMI ID 的设置。创建计算环境时，选择 AMI ID 的优先级首先是启动模板，然后是imageId设置，最后是imageIdOverride设置。但是，如果使用的 AMI ID 来自启动模板，则更新imageId或imageIdOverride设置都不会更新 AMI ID。更新从启动模板中选择的 AMI ID 的唯一方法是更新启动模板。如果启动模板的版本参数为$Default或$Latest，则会评估指定启动模板的默认版本或最新版本。如果默认选择了不同的 AMI ID 或选择了最新版本的启动模板，则将在更新中使用该 AMI ID。

如果未使用启动模板来选择 AMI ID，则会使用imageId或imageIdOverride参数中指定的 AMI ID。如果同时指定了这两个参数，则会使用imageIdOverride参数中指定的 AMI ID。

假设计算环境使用由imageId、imageIdOverride或launchTemplate参数指定的 AMI ID，并且要使用由AWS Batch支持的最新 Amazon ECS 优化 AMI。然后，更新必须删除提供 AMI ID 的设置。对于imageId，需要为该参数指定一个空字符串。对于imageIdOverride，需要为ec2Configuration参数指定一个空字符串。

如果 AMI ID 来自启动模板，则可以通过以下任一方法更改为AWS Batch支持的最新 Amazon ECS 优化 AMI：

- 通过为launchTemplateId或launchTemplateName参数指定一个空字符串，删除启动模板。这将删除整个启动模板，而不仅仅是 AMI ID。
- 如果启动模板的更新版本未指定 AMI ID，则必须将updateToLatestImageVersion参数设置为true。

Amazon EKS 计算环境

亚马逊 EK AWS Batch S 入门提供了创建 EKS 计算环境的简短指南。本节提供有关 Amazon EKS 计算环境的更多详细信息。

主题

默认 AMI 选择

创建 Amazon EKS 计算环境时，无需指定亚马逊系统映像 (AMI)。 AWS Batch 根据您的CreateComputeEnvironment请求中指定的Kubernetes版本和实例类型选择 Amazon EKS 优化的 AMI。一般情况下，我们建议您使用默认 AMI 选择。有关 Amazon EKS 优化的 AMI 的更多信息，请参阅 《Amazon EKS 用户指南》中的 Amazon EKS 优化 Amazon Linux AMI。

运行以下命令，查看为您的 Amazon EKS 计算环境选择了 AWS Batch 哪种 AMI 类型。以下示例是非 GPU 实例类型。

```
# compute CE example: indicates Batch has chosen the AL2 x86 or ARM EKS 1.29 AMI,
 depending on instance types
$ aws batch describe-compute-environments --compute-environments My-Eks-CE1 \
    | jq '.computeEnvironments[].computeResources.ec2Configuration'
[
  {
    "imageType": "EKS_AL2",
    "imageKubernetesVersion": "1.29"
  }
]
```

以下示例是 GPU 实例类型。

```
# GPU CE example: indicates Batch has choosen the AL2 x86 EKS Accelerated 1.29 AMI
```

```
$ aws batch describe-compute-environments --compute-environments My-Eks-GPU-CE \
    | jq '.computeEnvironments[].computeResources.ec2Configuration'
[
  {
    "imageType": "EKS_AL2_NVIDIA",
    "imageKubernetesVersion": "1.29"
  }
]
```

支持的Kubernetes版本

AWS Batch 在 Amazon 上，EKS 目前支持以下Kubernetes版本：

- 1.29
- 1.28
- 1.27
- 1.26
- 1.25
- 1.24
- 1.23

当使用 CreateComputeEnvironment API 操作或 UpdateComputeEnvironmentAPI 操作以创建或更新计算环境时，可能会看到类似于以下内容的错误消息。如果在EC2Configuration中指定不受支持的Kubernetes版本，则会出现此问题。

```
At least one imageKubernetesVersion in EC2Configuration is not supported.
```

要解决此问题，请删除计算环境，然后使用支持的Kubernetes版本重新创建。

可以在 Amazon EKS 集群上执行次要版本升级。例如，即使不支持次要版本，也可以将集群从1.xx升级到1.yy。

但是，主要版本更新后，计算环境的状态可能会更改为INVALID。例如，如果将主要版本从1.xx升级到2.yy。如果不支持主要版本 AWS Batch，则会看到类似于以下内容的错误消息。

```
reason=CLIENT_ERROR - ... EKS Cluster version [2.yy] is unsupported
```

更新计算环境的Kubernetes版本

使用 AWS Batch，您可以更新计算环境的Kubernetes版本以支持 Amazon EKS 集群升级。计算环境的Kubernetes版本是 AWS Batch 启动运行任务的Kubernetes节点的 Amazon EKS AMI 版本。在更新 Amazon EKS 集群的控制平面Kubernetes版本之前或之后，您可以在他们的 Amazon EKS 节点上执行版本升级。我们建议在升级控制面板后更新节点。有关更多信息，请参阅《Amazon EKS 用户指南》中的更新 Amazon EKS 集群Kubernetes版本。

要升级计算环境的Kubernetes版本，请使用 UpdateComputeEnvironmentAPI 操作。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources \
      'ec2Configuration=[{imageType=EKS_AL2,imageKubernetesVersion=1.23}]'
```

Kubernetes节点的共同责任

计算环境的维护是一项共同责任。

- 请勿更改或移除 AWS Batch 节点、标签、污点、命名空间、启动模板或 auto Scaling 组。不要向 AWS Batch 托管节点添加污点。如果要进行上述任何更改，则无法支持计算环境，并且会出现故障，包括空闲实例。
- 不要将 pod 定位到 AWS Batch 托管节点。如果将容器组（pod）定位到托管节点，则会出现扩展中断和作业队列卡死的情况。运行不在自管节点或托管节点组 AWS Batch 上使用的工作负载。有关更多信息，请参阅《Amazon EKS 用户指南》中的托管节点组。
- 您可以将 a 定位DaemonSet为在 AWS Batch 托管节点上运行。有关更多信息，请参阅 DaemonSet 在 AWS Batch 托管节点上运行。

AWS Batch 不会自动更新计算环境 AMI。您要负责更新它们。运行以下命令将 AMI 更新为最新 AMI 版本。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources 'updateToLatestImageVersion=true'
```

AWS Batch 不会自动升级Kubernetes版本。运行以下命令可将计算机环境Kubernetes版本更新到 *1.23*。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources \
      'ec2Configuration=[{imageType=EKS_AL2,imageKubernetesVersion=1.23}]'
```

在更新到最新的 AMI 或Kubernetes版本时，可以指定是否在作业更新时终止作业 (`terminateJobsOnUpdate`)，以及运行中的作业未完成的话要等待多长时间才替换实例 (`jobExecutionTimeoutMinutes`)。有关更多信息，请参阅更新计算环境和 UpdateComputeEnvironmentAPI 操作中设置的基础设施更新政策 (UpdatePolicy)。

DaemonSet在 AWS Batch 托管节点上运行

AWS Batch 在 AWS Batch 托管Kubernetes节点上设置污点。您可以通过以下方式DaemonSet将 a 设置为在 AWS Batch 托管节点上运行`tolerations`。

```
tolerations:
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"
```

执行此操作的另一种方法是使用以下`tolerations`。

```
tolerations:
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"
  effect: "NoSchedule"
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"
  effect: "NoExecute"
```

使用启动模板进行自定义

AWS Batch 在 Amazon 上，EKS 支持启动模板。启动模板的功能受到限制。

> **Important**
>
> AWS Batch 运行`/etc/eks/bootstrap.sh`。请勿在启动模板或cloud-inituser-data脚本中运行`/etc/eks/bootstrap.sh`。除了 bootstrap.sh 的`--kubelet-extra-args`参数外，还可以添加其他参数。为此，请在`AWS_BATCH_KUBELET_EXTRA_ARGS`文件中设置`/etc/aws-batch/batch.config`变量。详情请参阅以下示例。

> **Note**
>
> 如果在调用后CreateComputeEnvironment更改了启动模板，则UpdateComputeEnvironment必须调用该启动模板来评估要替换的启动模板的版本。

主题

- 添加kubelet额外参数
- 配置容器运行时系统
- 安装 Amazon EFS 卷
- IPv6 支持

添加kubelet额外参数

AWS Batch 支持向kubelet命令添加额外的参数。有关支持的参数列表，请参阅Kubernetes文档中的kubelet。在以下示例中，*--node-labels mylabel=helloworld*已添加到kubelet命令行中。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
mkdir -p /etc/aws-batch

echo AWS_BATCH_KUBELET_EXTRA_ARGS=\"--node-labels mylabel=helloworld\" >> /etc/aws-
batch/batch.config

--==MYBOUNDARY==--
```

配置容器运行时系统

您可以使用 AWS Batch CONTAINER_RUNTIME环境变量在托管节点上配置容器运行时系统。以下示例将容器运行时系统设置为“bootstrap.sh运行时containerd”。有关更多信息，请参阅Kubernetes文档中的containerd。

> **Note**
>
> CONTAINER_RUNTIME环境变量等同于bootstrap.sh的--container-runtime选项。有关更多信息，请参阅Kubernetes文档中的Options。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
mkdir -p /etc/aws-batch

echo CONTAINER_RUNTIME=containerd >> /etc/aws-batch/batch.config

--==MYBOUNDARY==--
```

安装 Amazon EFS 卷

可以使用启动模板将卷装载到节点上。在以下示例中，使用了cloud-configpackages和runcmd设置。有关更多信息，请参阅cloud-init文档中的云配置示例。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

packages:
- amazon-efs-utils

runcmd:
- file_system_id_01=fs-abcdef123
- efs_directory=/mnt/efs

- mkdir -p ${efs_directory}
- echo "${file_system_id_01}:/ ${efs_directory} efs _netdev,noresvport,tls,iam 0 0"
 >> /etc/fstab
- mount -t efs -o tls ${file_system_id_01}:/ ${efs_directory}
```

```
--==MYBOUNDARY==--
```

要在作业中使用此卷，必须将其添加到 eksPro perties 参数中。RegisterJobDefinition以下示例是作业定义的一大部分。

```
{
    "jobDefinitionName": "MyJobOnEks_EFS",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "containers": [
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": ["ls", "-la", "/efs"],
                    "resources": {
                        "limits": {
                            "cpu": "1",
                            "memory": "1024Mi"
                        }
                    },
                    "volumeMounts": [
                        {
                            "name": "efs-volume",
                            "mountPath": "/efs"
                        }
                    ]
                }
            ],
            "volumes": [
                {
                    "name": "efs-volume",
                    "hostPath": {
                        "path": "/mnt/efs"
                    }
                }
            ]
        }
    }
}
```

在节点中，Amazon EFS 卷装载在/mnt/efs目录中。在 Amazon EKS 作业的容器中，卷安装在/efs目录中。

IPv6 支持

AWS Batch 支持具有 IPv6 地址的 Amazon EKS 集群。无需自定义即可获得 AWS Batch 支持。但是，在开始之前，我们建议查看《Amazon EKS 用户指南》中为容器组（pod）和服务分配 IPv6 地址中概述的注意事项和条件。

计算资源内存管理

当 Amazon ECS 容器代理将计算资源注册到计算环境中时，代理必须确定计算资源可为作业保留的内存量。由于平台内存开销和系统内核占用的内存，此数量不同于 Amazon EC2 实例的已安装内存量。例如，`m4.large` 实例具有 8GiB 的已安装内存。但是，当计算资源注册时，这不总是表示确实有 8192 MiB 内存可用于作业。

假设您为作业指定了 8192 MiB，并且您的计算资源中没有一个有 8192 MiB 或更大的可用内存来满足此要求。则作业就无法放置在您的计算环境中。如果使用托管计算环境，则 AWS Batch 必须启动更大的实例类型来满足该要求。

原定设置的 AWS Batch 计算资源 AMI 还会为 Amazon ECS 容器代理以及其他关键系统进程预留 32 MiB 内存。此内存不能用于作业分配。有关更多信息，请参阅预留系统内存。

Amazon ECS 容器代理使用 Docker `ReadMemInfo()` 函数来查询可用于操作系统的总内存。Linux 提供了用来确定总内存的命令行实用程序。

Example - 确定 Linux 总内存

free 命令可返回操作系统识别的总内存。

```
$ free -b
```

以下是运行经 Amazon ECS 优化的 Amazon Linux AMI 的 `m4.large` 实例的示例输出。

```
             total       used       free     shared    buffers     cached
Mem:    8373026816  348180480 8024846336      90112   25534464  205418496
-/+ buffers/cache:  117227520 8255799296
```

此实例的总内存为 8373026816 字节。这意味着有 7985 MiB 可用于执行任务。

预留系统内存

如果您的作业占用计算资源上的所有内存，则您的作业可能会与关键系统进程争夺内存，并可能引起系统故障。Amazon ECS 容器代理提供一个名为 `ECS_RESERVED_MEMORY` 的配置变量。您可以使用

该配置变量从分配给您的作业的池中移除指定 MiB 数的内存。这可以有效地为关键系统进程预留该内存。

原定设置的 AWS Batch 计算资源 AMI 还会为 Amazon ECS 容器代理以及其他关键系统进程预留 32 MiB 内存。

查看计算资源内存

您可以在 Amazon ECS 控制台中或使用 DescribeContainerInstances API 操作查看计算资源注册的内存量。如果要尝试为作业提供尽可能多的某个具体实例类型的内存，以最大程度地提高资源使用率，则可以观察可用于该计算资源的内存，然后为作业分配该内存量。

查看计算资源内存

1. 在 https://console.aws.amazon.com/ecs/v2 打开控制台。
2. 选择 Clusters（集群），并选择托管您的计算资源的集群。

   计算环境的集群名称以该计算环境名称开头。
3. 选择基础架构。
4. 在容器实例下，选择容器实例。
5. 资源和网络连接部分 显示了计算资源的已注册内存和可用内存。

   已注册 内存值是计算资源首次启动时向 Amazon ECS 注册的值，可用 内存值是尚未分配给作业的值。

Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项

在 Amazon EKS 上的 AWS Batch 中，您可以指定可供容器使用的资源。例如，您可以为 vCPU 和内存资源指定 `requests` 或 `limits` 值。

以下是指定 vCPU 资源的限制：

- 至少必须指定一个 vCPU`requests` 或 `limits`。
- 一个 vCPU 单元等同于一个物理或虚拟内核。
- vCPU 值必须以整数或以 0.25 为增量输入。
- 最小的有效 vCPU 值为 0.25。
- 如果指定了两者的值，则 `requests` 值必须小于或等于 `limits` 值。这样，您就可以配置软和硬 vCPU 配置。

- 无法以 milliCPU 形式指定 vCPU 值。例如，100m 不是有效值。
- AWS Batch 使用 requests 值进行缩放决策。如果未指定 requests 值，则会将 limits 值复制到 requests 值中。

以下是指定内存资源的限制：

- 至少必须指定内存 requests 或 limits 值之一。
- 内存值必须在 mebibytes (MiBs) 中。
- 如果两者都指定，则 requests 值必须等于 limits 值。
- AWS Batch 使用 requests 值进行缩放决策。如果未指定 requests 值，则会将 limits 值复制到 requests 值中。

以下是指定 GPU 资源的限制：

- 如果两者都指定，则 requests 值必须等于 limits 值。
- AWS Batch 使用 requests 值进行缩放决策。如果未指定 requests 值，则会将 limits 值复制到 requests 值中。

作业定义示例

Amazon EKS 作业定义的以下 AWS Batch 配置了软 vCPU 共享。这允许在 Amazon EKS 上的 AWS Batch 使用该实例类型的所有 vCPU 容量。但是，如果还有其他作业在运行，则会为该作业分配最多 2 vCPU。内存限制为不超过 2 GB。

```
{
    "jobDefinitionName": "MyJobOnEks_Sleep",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "containers": [
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": ["sleep", "60"],
                    "resources": {
                        "requests": {
                            "cpu": "2",
                            "memory": "2048Mi"
```

```
                        }
                    }
                }
            ]
        }
    }
}
```

Amazon EKS 作业定义上的以下 AWS Batch 的 1 值为 `request` ，最多可为该作业分配 4 vCPU。

```
{
    "jobDefinitionName": "MyJobOnEks_Sleep",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "containers": [
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": ["sleep", "60"],
                    "resources": {
                        "requests": {
                            "cpu": "1"
                        },
                        "limits": {
                            "cpu": "4",
                            "memory": "2048Mi"
                        }
                    }
                }
            ]
        }
    }
}
```

Amazon EKS 任务定义上的以下 AWS Batch 将 vCPU `limits` 值设置为 1，内存 `limits` 值设置为 1 GB。

```
{
    "jobDefinitionName": "MyJobOnEks_Sleep",
    "type": "container",
    "eksProperties": {
        "podProperties": {
            "containers": [
```

```
                {
                    "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                    "command": ["sleep", "60"],
                    "resources": {
                        "limits": {
                            "cpu": "1",
                            "memory": "1024Mi"
                        }
                    }
                }
            ]
        }
    }
}
```

在 AWS Batch 将 Amazon EKS 作业上的 AWS Batch 转换为 Amazon EKS 容器组（pod）时，AWS Batch 会将 `limits` 值复制给 `requests` 值。在未指定 `requests` 值时，会出现这种情况。提交前面的示例作业定义时，容器组（pod）`spec` 如下所示。

```
apiVersion: v1
kind: Pod
...
spec:
  ...
  containers:
    - command:
        - sleep
        - 60
      image: public.ecr.aws/amazonlinux/amazonlinux:2
      resources:
        limits:
          cpu: 1
          memory: 1024Mi
        requests:
          cpu: 1
          memory: 1024Mi
      ...
```

节点 CPU 和内存预留

AWS Batch 依赖 `bootstrap.sh` 文件的默认逻辑来预留 vCPU 和内存。有关 `bootstrap.sh` 文件的更多信息，请参阅 bootstrap.sh。在调整 vCPU 和内存资源的大小时，请考虑以下示例。

> **Note**
>
> 如果没有实例在运行，vCPU 和内存预留最初可能会影响 AWS Batch 扩展逻辑和决策。实例运行后，AWS Batch 调整初始分配。

节点 CPU 预留示例

CPU 预留值是使用实例可用的 vCPU 总数以毫核为单位计算的。

vCPU 数量	预留百分比
1	6%
2	1%
3-4	0.5%
4 及以上	0.25%

使用上述值，以下各项为真：

- 具有 2 个 vCPU 的 `c5.large` 实例的 CPU 预留值为 70 m。计算方法如下：(1*60) + (1*10) = 70 m。
- 具有 96 个 vCPU 的 `c5.24xlarge` 实例的 CPU 预留值为 310 m。计算方法如下：(1*60) + (1*10) + (2*5) + (92*2.5) = 310 m。

在此示例中，有 1930 个（计算为 2000-70）毫核 vCPU 单元可用于在 `c5.large` 实例上运行作业。假设您的作业需要 2 (2*1000 m) 个 vCPU 单元，则该作业不适合单个 `c5.large` 实例。但是，需要 `1.75` vCPU 单元的作业合适。

节点内存预留示例

内存预留值以 MB 为单位计算，使用以下公式：

- 实例容量以 MB 为单位。例如，一个 8 GB 实例为 7,748 MiB。

- `kubeReserved` 值。`kubeReserved` 值是为系统进程守护程序保留的内存量。`kubeReserved` 值的计算方式如下：((11 * 实例类型支持的最大容器组（pod）数) + 255)。有关实例类型支持的最大容器组（pod）数量的信息，请参阅 eni-max-pods.txt
- `HardEvictionLimit` 值。当可用内存低于 `HardEvictionLimit` 值时，实例会尝试驱逐容器组（pod）。

计算可分配内存的公式如下：(*instance_capacity_in_MiB*) - (11 * (*maximum_number_of_pods*)) - 255 - (*HardEvictionLimit value.*))。

一个 `c5.large` 实例最多支持 29 个容器组（pod）。对于 `HardEvictionLimit` 值为 100 MiB 的 8 GB `c5.large` 实例，可分配的内存为 7074 MiB。这是通过以下方式计算的：(7748 - (11 * 29) -255 -100) = 7074 MiB。在此示例中，8,192 MiB 作业不适合此实例，尽管它是 8 gibibyte (GiB) 实例。

DaemonSets

在使用 DaemonSets 时，请考虑以下几点：

- 如果 Amazon EKS 实例上没有 AWS Batch 在运行，则 DaemonSets 最初可能会影响 AWS Batch 扩展逻辑和决策。AWS Batch 最初为预期 DaemonSets 分配了 0.5 个 vCPU 单元和 500 MiB。实例运行后，AWS Batch 调整初始分配。
- 如果 DaemonSet 定义了 vCPU 或内存限制，那么在 Amazon EKS 作业上的 AWS Batch 的资源就会减少。我们建议您将分配给 AWS Batch 作业 DaemonSets 的数量保持在尽可能低的水平。

计划策略

您可以使用调度策略来配置如何在用户或工作负载之间分配作业队列中的计算资源。使用计划策略，您可以为工作负载或用户分配不同的公平份额标识符。AWS Batch 为每个公平份额标识符分配一段时间内可用资源总额的百分比。

公平份额百分比是使用 `shareDecaySeconds` 和 `shareDistribution` 值计算的。您可以通过为策略分配份额衰减时间来增加公平份额分析的时间。增加时间会增加时间的权重，而减少定义的权重。您可以通过指定计算预留来保留计算资源，以备不活跃的公平份额标识符使用。有关更多信息，请参阅计划策略参数。

主题

创建计划策略

在创建带有计划策略的任务队列之前，您必须创建计划策略。创建计划策略时，您可以将一个或多个公平份额标识符或公平份额标识符前缀与队列的权重相关联，还可以选择为策略分配衰减周期和计算预留。

要创建计划策略

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择创建策略和创建。
4. 对于 Name，请为您的计划策略输入唯一的名称。最多能包含 128 个字母（大写和小写字母）、数字、连字符和下划线。
5. （可选）在共享衰减秒数中，输入计划策略的份额衰减时间的整数值。较长的共享衰减时间将需要在调度作业时考虑较长时间内的计算资源使用量。如果公平份额标识符最近没有使用计算资源，则这可能允许使用公平份额标识符的作业暂时使用超过该公平份额标识符的权重所允许的计算资源。
6. （可选）在计算预留中，为计划策略的计算预留输入一个整数值。计算预留将保留一些 vCPU，用于当前未处于活动状态的公平份额标识符。

   预留比为 `(`*`computeReservation`*`/100)^`*`ActiveFairShares`*，其中 ActiveFairShares 是活动公平份额标识符的数量。

例如，computeReservation 值为 50 表示，如果只有一个公平份额标识符，则 AWS Batch 应预留最大可用 VCPU 的 50%；如果有两个公平份额标识符，则应预留 25%；如果有三个公平份额标识符，则应预留 12.5%。computeReservation 值为 25 表示，如果只有一个公平份额标识符，则 AWS Batch 应预留最大可用 VCPU 的 25%；如果有两个公平份额标识符，则应预留 6.25%；如果有三个公平份额标识符，则应预留 1.56%。

7. 在份额属性部分中，您可以为要与计划策略关联的每个公平份额标识符指定公平份额标识符和权重。

   a. 选择添加份额标识符。

   b. 在份额标识符中，指定公平份额标识符。如果字符串以 '*' 结尾，则它将成为公平份额标识符前缀，用于匹配作业的公平份额标识符。调度策略中的所有公平份额标识符和公平份额标识符前缀都必须是唯一的，不能重叠。例如，您不能在同一个计划策略中使用公平份额标识符前缀“UserA*”和公平份额标识符“UserA1”。

   c. 在权重系数中，指定公平份额标识符的相对权重。默认值为 1.0。值越低，计算资源的优先级越高。如果使用公平份额标识符前缀，则具有以该前缀开头的公平份额标识符的作业将共享权重系数。这实际上增加了这些作业的权重系数，降低了它们各自的优先级，但公平份额标识符前缀的权重系数保持不变。

8. （可选）在标签部分中，可以指定要与计划策略关联的每个标签的键和值。有关更多信息，请参阅对AWS Batch资源加标签。

9. 选择提交以完成并创建您的计划策略。

计划策略模板

下面显示了一个空的计划策略模板。您可以使用此模板创建计划策略，随后可将计划策略保存到文件并与 AWS CLI --cli-input-json 选项结合使用。有关这些参数的更多信息，请参阅 AWS BatchAPI 参考 中的 CreateSchedulingPolicy。

```
{
    "name": "",
    "fairsharePolicy": {
        "shareDecaySeconds": 0,
        "computeReservation": 0,
        "shareDistribution": [
            {
                "shareIdentifier": "",
                "weightFactor": 0.0
            }
```

```
        ]
    },
    "tags": {
        "KeyName": ""
    }
}
```

> **Note**
>
> 您可以使用以下 AWS CLI 命令生成前述作业队列模板。
>
> ```
> $ aws batch create-scheduling-policy --generate-cli-skeleton
> ```

计划策略参数

计划策略分为三个基本组成部分：计划策略的名称、公平份额策略和标签。

主题

- 计划策略名称
- 公平份额策略
- 标签

计划策略名称

`name`

计划策略的名称。最多能包含 128 个字母（大写和小写字母）、数字、连字符和下划线。

类型：字符串

必需：是

公平份额策略

`fairsharePolicy`

计划策略的公平份额策略。

```
"fairsharePolicy": {
   "computeReservation": number,
   "shareDecaySeconds": number,
   "shareDistribution": [
      {
         "shareIdentifier": "string",
         "weightFactor": number
      }
   ]
}
```

类型：对象

必需：否

computeReservation

一个值，用于为尚未使用的公平份额标识符预留一些可用的最大 VCPU。

预留比为 (*computeReservation*/100)^*ActiveFairShares*，其中 ActiveFairShares 是活动公平份额标识符的数量。

例如，computeReservation 值为 50 表示，如果只有一个活跃的公平份额标识符，则 AWS Batch 应预留最大可用 VCPU 的 50%；如果有两个活跃的公平份额标识符，则应预留 25%；如果有三个活跃的公平份额标识符，则应预留 12.5%。computeReservation 值为 25 表示，如果只有一个活跃的公平份额标识符，则 AWS Batch 应预留最大可用 VCPU 的 25%；如果有两个活跃的公平份额标识符，则应预留 6.25%；如果有三个活跃的公平份额标识符，则应预留 1.56%。

类型：整数

有效范围：最小值为 0。最大值为 99。

必需：否

shareDecaySeconds

用于计算每个正在使用的公平份额标识符的公平份额百分比的时间段。值为零（0）表示仅测量当前使用量。衰减使最近运行的作业比之前运行的作业具有更高的权重。

类型：整数

有效范围：最小值为 0。最大值为 604800（1 周）。

必需：否

shareDistribution

对象的数组，其中包含公平份额策略的公平份额标识符的权重。未包含的公平份额标识符的默认权重为 1.0。

```
"shareDistribution": [
   {
      "shareIdentifier": "string",
      "weightFactor": number
   }
]
```

类型：数组

必需：否

shareIdentifier

公平份额标识符或公平份额标识符前缀。如果字符串以“*”结尾，则此字符串为以该前缀开头的公平份额标识符指定公平份额标识符前缀。例如，如果值为 UserA*，weightFactor 为 1，并且有两个以 UserA 开头的公平份额标识符，则每个公平份额标识符的权重都将为 2；如果有五个这样的公平份额标识符，则每个标识符的权重都将为 5。

公平份额策略中的公平份额标识符以及公平份额标识符前缀列表不能重叠。例如，在同一公平份额策略中，无法同时存在公平份额标识符的前缀为 UserA* 且公平份额标识符为 UserA-1 。

类型：字符串

必需：是

weightFactor

公平份额标识符的权重系数。默认值为 1.0。值越低，计算资源的优先级越高。例如，使用权重因子为 0.125（1/8）的共享标识符的作业获得的计算资源是使用权重因子为 1 的共享标识符的作业的 8 倍。

支持的最小值为 0.0001,，支持的最大值为 999.9999。

类型：浮点值

必需：否

标签

`tags`

与计划策略关联的键值对标签。有关更多信息，请参阅对AWS Batch资源加标签。

类型：字符串到字符串映射

必需：否

在AWS Batch控制台中使用 Step Functions 状态机编排AWS Batch作业

可以使用AWS Batch控制台查看有关 Step Functions 状态机及其使用的功能的详细信息。

章节

查看状态机详细信息

AWS Batch控制台会显示当前AWS 区域中至少包含一个提交AWS Batch作业的工作流步骤的状态机列表。

选择状态机可查看工作流的图示。以蓝色突出显示的步骤表示AWS Batch作业。使用图形控件将图形放大、缩小和居中放置。

> **Note**
>
> 在状态机定义中使用 JSONPath AWS Batch动态引用作业时，AWS Batch控制台中将无法显示功能详细信息。相反，作业名称将作为动态引用列出，并且图示中的相应步骤也会显示为灰色。

查看状态机详细信息

1. 打开由 Step Functions 提供支持的AWS Batch控制台工作流程编排页面。
2. 选择状态机。
   <result>

   AWS Batch控制台会打开详细信息页面。
   </result>

有关更多信息，请参阅 AWS Step Functions开发人员指南中的 Step Functions。

编辑状态机

如果要编辑状态机，AWS Batch会打开 Step Functions 控制台的编辑定义页面。

要编辑状态机

1. 打开由 Step Functions 提供支持的AWS Batch控制台工作流程编排页面。
2. 选择状态机。
3. 选择编辑。

   Step Functions 控制台会打开编辑定义页面。
4. 编辑状态机，然后选择保存。

有关编辑状态机的更多信息，请参阅AWS Step Functions开发人员指南中的 Step Function 状态机语言。

运行状态机

如果要运行状态机，AWS Batch会打开 Step Functions 控制台的新执行页面。

要运行状态机

1. 打开由 Step Functions 提供支持的AWS Batch控制台工作流程编排页面。
2. 选择状态机。
3. 选择执行。

   Step Functions 控制台会打开新执行页面。
4. （可选）编辑状态机，然后选择开始执行。

有关运行状态机的更多信息，请参阅AWS Step Functions 开发人员指南中的 Step Functions 状态机执行概念。

AWS Batch 在 AWS Fargate

AWS Fargate 是可与 AWS Batch 结合使用的技术，使您在运行容器时不必管理服务器或Amazon EC2 实例的集群。使用 AWS Fargate，您不必再预配置、配置或扩展虚拟机集群即可运行容器。这样一来，您就无需再选择服务器类型、确定扩展集群的时间和优化集群打包。

您在运行使用 Fargate 启动类型的任务和服务时，您需要将应用程序打包到容器中、指定 CPU 和内存要求、定义联网和 IAM policy 并启动应用程序。每个 Fargate 任务都具有自己的隔离边界，不与其他任务共享底层内核、CPU 资源、内存资源或弹性网络接口。

目录

何时使用 Fargate

我们建议在大多数情况下使用 Fargate。Fargate 启动并扩展计算以密切匹配您为容器指定的资源需求。有了 Fargate，您无需过度配置或为额外的服务器付费。您也不必担心与基础设施相关的参数（例如实例类型）的细节。当计算环境需要扩展时，可以更快地启动在 Fargate 资源上运行的作业。通常需要几分钟才能启动新的 Amazon EC2 实例。但是，在 Fargate 上运行的作业可以在大约 30 秒内完成配置。所需的确切时间取决于多个因素，包括容器映像大小和作业数量。

但是，如果您的作业需要下列任一条件，我们建议您使用 Amazon EC2：

- 超过 16 个 vCPU
- 超过 120 吉字节 (GiB) 的内存
- 一个 GPU
- 使用自定义亚马逊机器映像（AMI）
- 任何 linuxParameters 参数

如果您有大量的作业，我们建议您使用 Amazon EC2 基础架构。例如，如果同时运行的作业数量超过了 Fargate 的节流限制。这是因为，使用 EC2，向 EC2 资源分配作业的速率要高于 Fargate 资源。

此外，当您使用 EC2 时，可以同时运行更多作业。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 AWS Fargate 服务限额。

Fargate 上的作业定义

Fargate 上的 AWS Batch 任务并不支持所有可用的任务定义参数。某些参数完全不受支持，而其他参数对于 Fargate 任务的行为则不同。

以下列表描述了在 Fargate 作业中无效或以其他方式受到限制的作业定义参数。

`platformCapabilities`

必须指定为 `FARGATE`。

```
"platformCapabilities": [ "FARGATE" ]
```

`type`

必须指定为 `container`。

```
"type": "container"
```

`containerProperties` 中的参数

`executionRoleArn`

对于在 Fargate 资源上运行的作业，指定。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的任务的 IAM 角色。

```
"executionRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole"
```

`fargatePlatformConfiguration`

（可选，仅适用于 Fargate 作业定义）。指定 Fargate 平台版本或 `LATEST` 最新平台版本。`platformVersion` 的可能值为 `1.3.0`、`1.4.0` 和 `LATEST`。

```
"fargatePlatformConfiguration": { "platformVersion": "1.4.0" }
```

`instanceType`, `ulimits`

不适用于在 Fargate 资源上运行的作业。

`memory`, `vcpus`

这些设置必须在 `resourceRequirements` 中指定

`privileged`

要么不指定此参数，要么指定 `false`。

```
"privileged": false
```

`resourceRequirements`

必须使用支持的值来指定内存和 vCPU 要求。GPU 资源在 Fargate 资源上运行的作业不支持 GPU 资源。

如果您使用 GuardDuty 运行时监控，则 GuardDuty 安全代理会有轻微的内存开销。因此，内存限制必须包括 GuardDuty安全代理的大小。有关 GuardDuty 安全代理内存限制的信息，请参阅《GuardDuty 用户指南》中的 CPU 和内存限制。有关最佳实践的信息，请参阅 Amazon ECS 开发人员指南中的启用运行时监控后如何修复 Fargate 任务中的内存不足错误。

```
"resourceRequirements": [
  {"type": "MEMORY", "value": "512"},
  {"type": "VCPU",   "value": "0.25"}
]
```

`linuxParameters` 中的参数

`devices`, `maxSwap`, `sharedMemorySize`, `swappiness`, `tmpfs`

不适用于在 Fargate 资源上运行的作业。

`logConfiguration` 中的参数

`logDriver`

仅支持 `awslogs` 和 `splunk`。有关更多信息，请参见 使用 awslogs 日志驱动程序。

`networkConfiguration` 中的会员

`assignPublicIp`

如果私有子网未连接用于向互联网发送流量的 NAT 网关，`assignPublicIp` 则必须为“ENABLED”。有关更多信息，请参见 AWS Batch 执行 IAM 角色。

Fargate 上的作业队列

Fargate 上的 AWS Batch 作业队列基本保持不变。唯一的限制是 `computeEnvironmentOrder` 中列出的计算环境必须全部是 Fargate 计算环境（`FARGATE` 或 `FARGATE_SPOT`）。EC2 和 Fargate 的计算环境不能混合使用。

Fargate 上的计算环境

Fargate 上的 AWS Batch 计算环境并非支持所有可用的计算环境参数。某些参数完全不受支持。其他则对 Fargate 有具体要求。

以下列表描述了在 Fargate 作业中无效或以其他方式受到限制的计算环境参数。

`type`

此参数必须设置为 `MANAGED`。

```
"type": "MANAGED"
```

`computeResources` 对象中的参数

`allocationStrategy`, `bidPercentage`, `desiredvCpus`, `imageId`, `instanceTypes`, `ec2Configuration`, `ec2KeyPair`, `instanceRole`, `launchTemplate`, `minvCpus`, `placementGroup`, `spotIamFleetRole`

它们不适用于 Fargate 计算环境，也无法提供。

`subnets`

如果此参数中列出的子网未连接 NAT 网关，则必须将作业定义中的 `assignPublicIp` 参数设置为 `ENABLED`。

`tags`

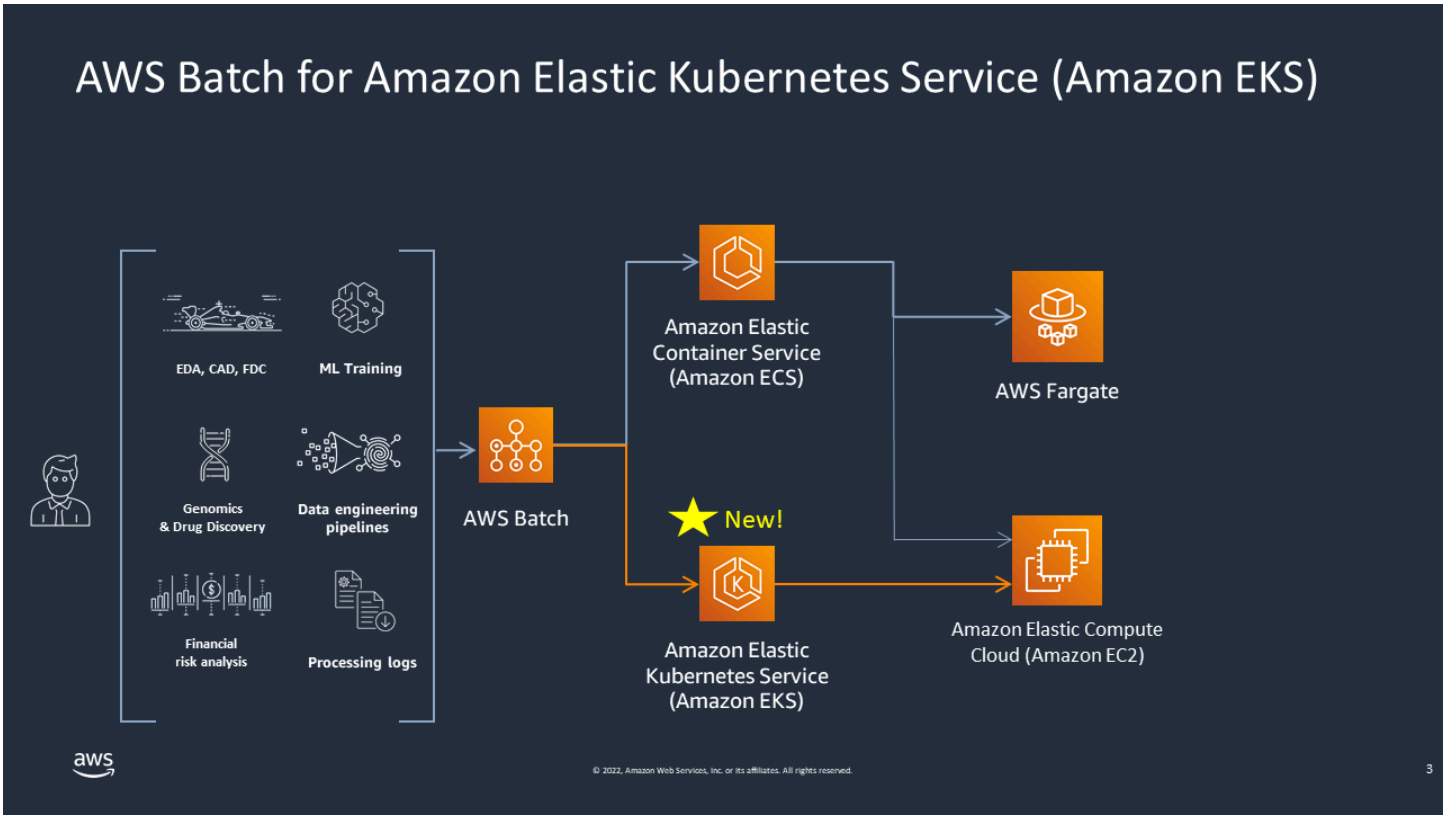
它们不适用于 Fargate 计算环境，也无法提供。要为 Fargate 计算环境指定标签，请使用 `computeResources` 对象中没有的 `tags` 参数。

`type`

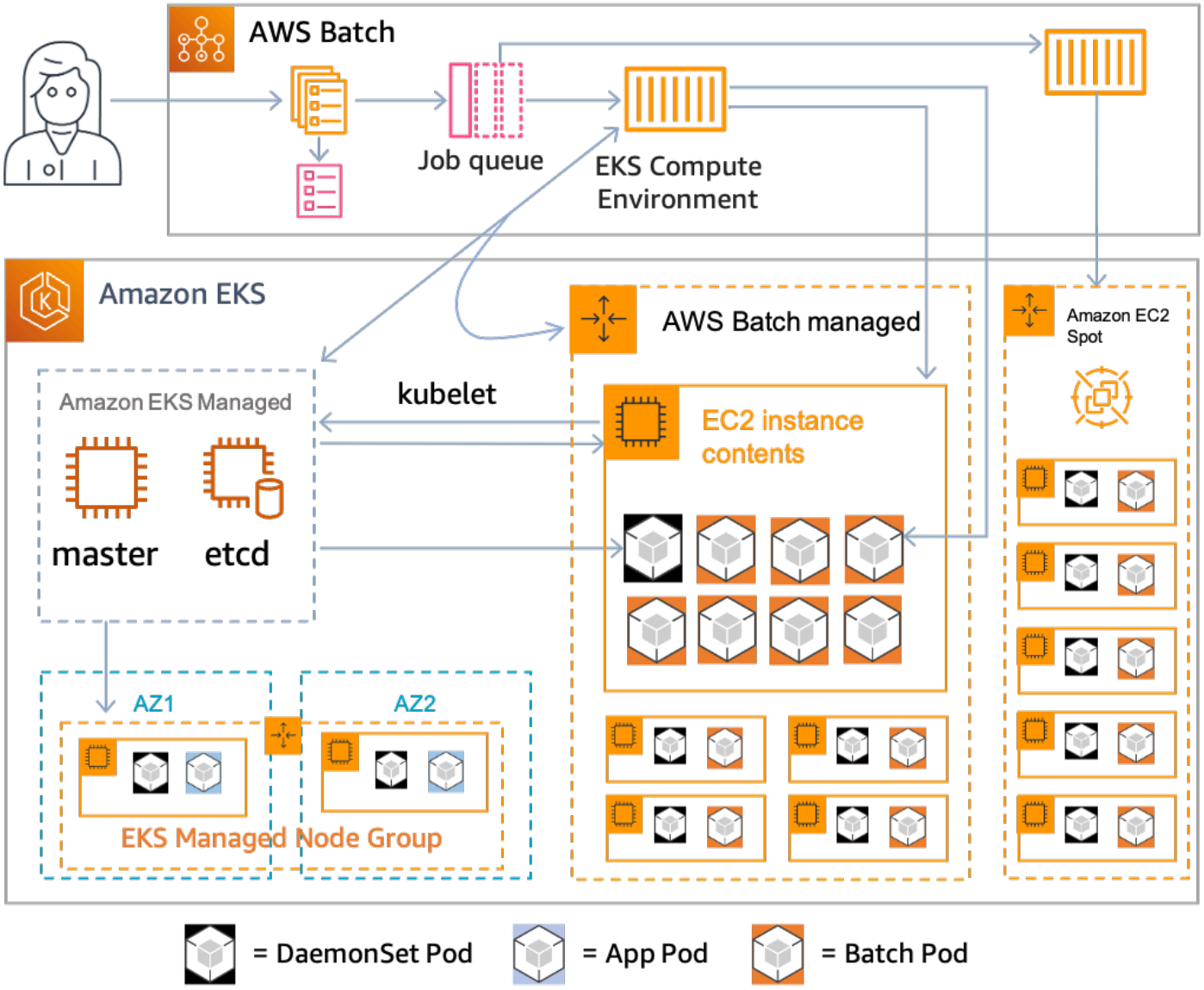
必须是 `FARGATE` 或 `FARGATE_SPOT`。

```
"type": "FARGATE_SPOT"
```

AWS Batch 在亚马逊 EKS 上



AWS Batch 通过提供托管批处理功能，简化 Amazon EKS 集群上的批处理工作负载。这包括队列、依赖关系跟踪、托管作业重试次数和优先级、Pod 管理和节点扩展。 AWS Batch 可以处理多个可用区和多个 Amazon EC2 实例类型和大小。 AWS Batch 集成了多个 Amazon EC2 Spot 最佳实践，以容错方式运行您的工作负载，从而减少中断。您可以使用 AWS Batch 来放心地运行少量夜间作业或数百万个关键任务作业。



AWS Batch 是一项托管服务，用于协调Kubernetes集群中的批量工作负载，这些工作负载由亚马逊 Elastic Kubernetes Service (Amazon EKS) 管理。 AWS Batch 使用 “叠加” 模型在集群外部进行这种编排。由于 AWS Batch 是托管服务，因此无需在集群中安装或管理任何Kubernetes组件（例如，操作员或自定义资源）。 AWS Batch 只需要将您的集群配置为允许 AWS Batch 与 API 服务器通信的基于角色的访问控制 (RBAC)。Kubernetes AWS Batch 调用 Kubernetes API 来创建、监控和删除 Kubernetes Pod 和节点。

AWS Batch 具有内置的扩展逻辑，可根据任务队列负载扩展Kubernetes节点，并在作业容量分配方面进行了优化。当任务队列为空时，将节点 AWS Batch 缩小到您设置的最小容量，默认情况下为零。AWS Batch 管理这些节点的整个生命周期，并用标签和污点装饰节点。这样，其他Kubernetes工作负载就不会放在由管理的节点上 AWS Batch。唯一的例外是`DaemonSets`，它可以将 AWS Batch 节点

作为目标，以提供正确执行作业所需的监控和其他功能。此外， AWS Batch 不会在集群中它不管理的节点上运行作业，特别是 pod。这样，您就可以为集群上的其他应用程序使用单独的扩展逻辑和服务。

要向提交作业 AWS Batch，您可以直接与 AWS Batch API 进行交互。 AWS Batch 将任务转换为 Amazon EKS 集群中由管理的节点，podspecs然后创建请求以将 Pod 放置在 Amazon EKS 集群 AWS Batch 中由管理的节点上。您可以使用诸如 kubectl 之类的工具查看正在运行的容器组（pod）和节点。当 Pod 完成执行后， AWS Batch 会删除其创建的 Pod 以保持较低的Kubernetes系统负载。

您可以先将有效的 Amazon EKS 集群与连接起来 AWS Batch。然后将 AWS Batch 任务队列附加到该队列，并使用podspec等效属性注册 Amazon EKS 任务定义。最后，使用引用作业定义的 SubmitJobAPI 操作提交作业。有关更多信息，请参阅 亚马逊 EK AWS Batch S 入门。

Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) 是一种用于加速高性能计算 (HPC) 应用程序的网络设备。如果满足以下条件，则 AWS Batch 支持使用 EFA 的应用程序。

- 有关支持 EFA 的实例类型列表，请参阅 Amazon EC2 用户指南中的支持的实例类型。

  > **Tip**
  >
  > 要在中查看支持 EFA 的实例类型列表 AWS 区域，请运行以下命令。然后，交叉引用 AWS Batch 控制台中返回的列表和可用实例类型列表。
  >
  > ```
  > $ aws ec2 describe-instance-types  --region us-east-1  --filters Name=network-info.efa-supported,Values=true  --query "InstanceTypes[*].[InstanceType]"  --output text | sort
  > ```

- 如需了解支持 EFA 的操作系统列表，请参阅支持的操作系统。
- AMI 加载了 EFA 驱动程序。
- EFA 的安全组必须允许进出安全组本身的所有入站和出站流量。
- 使用 EFA 的所有实例都必须位于同一集群置放群组中。
- 作业定义必须包含`devices`成员，其`hostPath`设置为`/dev/infiniband/uverbs0`，以允许将 EFA 设备传递到容器。如果指定了`containerPath`，则它还必须设置为`/dev/infiniband/uverbs0`。如果设置了`permissions`，则它必须设置为`READ` | `WRITE` |。`MKNOD`

  对于多节点 parallel 作业和单节点容器作业，LinuxParameters成员的位置不同。以下示例显示了差异，但缺少必填值。

  Example 多节点并行作业的示例

  ```
  {
    "jobDefinitionName": "EFA-MNP-JobDef",
    "type": "multinode",
    "nodeProperties": {
      ...
      "nodeRangeProperties": [
        {
          ...
          "container": {
  ```

```
          ...
          "linuxParameters": {
            "devices": [
              {
                "hostPath": "/dev/infiniband/uverbs0",
                "containerPath": "/dev/infiniband/uverbs0",
                "permissions": [
                    "READ", "WRITE", "MKNOD"
                ]
              },
            ],
          },
        },
      },
    ],
  },
}
```

Example 单节点容器作业的示例

```
{
  "jobDefinitionName": "EFA-Container-JobDef",
  "type": "container",
  ...
  "containerProperties": {
    ...
    "linuxParameters": {
      "devices": [
        {
          "hostPath": "/dev/infiniband/uverbs0",
        },
      ],
    },
  },
}
```

有关 EFA 的更多信息，请参阅 Amazon EC2 用户指南中的弹性结构适配器。

AWS Batch IAM policies、角色和权限

默认情况下，用户没有创建或修改 AWS Batch 资源或使用 AWS Batch API、AWS Batch 控制台或 AWS CLI 以执行任务的权限。要允许用户执行这些操作，请创建 IAM policy，授予用户使用特定资源和 API 操作的权限。然后，将这些策略附加到需要这些权限的用户或组。

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户对指定资源执行指定任务。有关更多信息，请参阅 IAM 用户指南中的权限与策略。有关管理和创建自定义 IAM policy 的更多信息，请参阅管理 IAM policy。

AWS Batch 代表您调用其他 AWS 服务。因此，AWS Batch 必须使用您的凭证进行身份验证。更具体地说，AWS Batch 通过创建 IAM 角色和提供这些权限的策略来进行身份验证。然后，它会在您创建它们时将角色与您的计算环境相关联。有关更多信息，请参阅《IAM 用户指南》中 Amazon ECS 实例角色、IAM 角色、使用服务相关角色和创建角色向 AWS 服务委派权限。

开始使用

IAM policy 必须授予或拒绝使用一个或多个 AWS Batch 操作的权限。

主题

策略结构

以下主题说明 IAM policy 的结构。

主题

- 策略语法
- AWS Batch 操作
- 适用于 AWS Batch 的Amazon 资源名称
- 检查用户是否具有所需权限

策略语法

IAM policy 是包含一个或多个语句的 JSON 文档。每个语句的结构如下。

```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
    "key":"value"
    }
      }
    }
  ]
}
```

组成语句的各个元素如下：

- Effect：此 effect 可以是 `Allow` 或 `Deny`。默认情况下 用户没有使用资源和 API 操作的权限。因此，所有请求都将被拒绝。显式允许将覆盖默认规则。显式拒绝将覆盖任何允许。
- Action：action 是对其授予或拒绝权限的特定 API 操作。有关如何指定操作的说明，请参阅 AWS Batch 操作。
- Resource：受操作影响的资源。有些 AWS Batch API 操作允许您在策略中包括该操作可以创建或修改的特定资源。要在语句中指定资源，您可使用其 Amazon 资源名称 (ARN)。有关更多信息，请参阅 AWS Batch API 操作支持的资源级权限 和 适用于 AWS Batch 的Amazon 资源名称。如果 AWS Batch API 操作目前不支持资源级权限，则使用通配符 (*) 来指定该操作可以影响的所有资源。
- Condition：条件是可选的。它们可以用于控制策略生效的时间。

有关 AWS Batch 的示例 IAM policy 的更多信息，请参阅 创建 AWS Batch IAM policy。

AWS Batch 操作

在 IAM policy 语句中，您可以从支持 IAM 的任何服务中指定任何 API 操作。对于 AWS Batch，请使用以下前缀为 API 操作命名：`batch:`（例如 `batch:SubmitJob` 和 `batch:CreateComputeEnvironment`）。

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": ["batch:action1", "batch:action2"]
```

您也可以使用通配符 (*) 指定多项操作。例如，您可以指定名称以单词“Describe”开头的所有操作。

```
"Action": "batch:Describe*"
```

要指定所有 AWS Batch API 操作，请包含通配符 (*)。

```
"Action": "batch:*"
```

有关 AWS Batch API 操作的列表，请参阅 AWS Batch API 参考 中的 操作。

适用于 AWS Batch 的Amazon 资源名称

每个 IAM policy 语句适用于您使用 Amazon 资源名称（ARN）指定的资源。

Amazon 资源名称（ARN）具有以下通用语法：

```
arn:aws:[service]:[region]:[account]:resourceType/resourcePath
```

service

服务 (例如，`batch`)。

区域

用于资源的 AWS 区域（例如 `us-east-2`）。

账户

AWS 账户 ID，不包含连字符（例如，`123456789012`）。

resourceType

资源类型 (例如，`compute-environment`)。

resourcePath

识别资源的路径。您可以在路径中使用通配符 (*)。

AWS Batch API 操作当前支持多个 API 操作的资源级权限。有关更多信息，请参阅 AWS Batch API 操作支持的资源级权限。要指定所有资源，或者如果特定 API 操作不支持 ARN，请在 `Resource` 元素中使用通配符 (*)。

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

检查用户是否具有所需权限

在实施 IAM policy 之前，请确保为用户授予使用其所需的特定 API 操作和资源的权限。

为此，首先创建一个用于测试目的的用户，然后将 IAM policy 附加到该测试用户。然后，以测试用户身份提出请求。您可以在控制台中提出测试请求，也可以使用 AWS CLI 提出测试请求。

> **Note**
>
> 您也可以使用 IAM Policy Simulator 测试您的策略。有关策略模拟器的更多信息，请参阅 IAM 用户指南中的使用 IAM Policy Simulator。

如果策略未向用户授予您所期望的权限，或者策略过度宽松，可以根据需要调整策略。重新测试，直到获得预期的结果。

> **Important**
>
> 在其生效之前，它需要几分钟时间将策略更改为适合状态。因此，我们建议您在测试策略更新前，等候至少五分钟的时间。

如果身份验证检查失败，该请求将返回一个带有诊断信息的代码消息。您可以使用 `DecodeAuthorizationMessage` 操作对消息进行解码。有关更多信息，请参阅 AWS Security Token Service API Reference 中的 DecodeAuthorizationMessage，以及 AWS CLI Command Reference 中的 decode-authorization-message。

AWS Batch API 操作支持的资源级权限

资源级权限 一词是指指定允许用户对哪些资源执行操作的能力。AWS Batch 对资源级权限提供部分支持。对于某些 AWS Batch 操作，您可以控制何时允许用户执行操作（基于必须满足的条件）。您还可以根据允许用户使用的特定资源进行控制。例如，您可以向用户授予提交作业的权限，但仅授予特定作业队列的权限，并且仅具有特定的作业定义。

下面的列表说明了支持资源级权限的 AWS Batch API 操作。列表还描述了每个操作支持的资源、资源 ARN 和条件键。

> **Important**
>
> 如果 AWS Batch API 操作未在此表中列出，则表示它不支持资源级权限。如果 AWS Batch API 操作不支持资源级权限，您可以为用户授予使用该操作的权限。但是，您必须为策略语句的资源元素添加通配符 (*)。

操作

CancelJob, CreateComputeEnvironment, CreateJobQueue, CreateSchedulingPolicy, DeleteComputeEnvironment, DeleteJobQueue, DeleteSchedulingPolicy, DeregisterJobDefinition, ListTagsForResource, RegisterJobDefinition, SubmitJob, TagResource, TerminateJob, UntagResource, UpdateComputeEnvironment, UpdateSchedulingPolicy, UpdateJobQueue

CancelJob

取消 AWS Batch 队列中的作业。

资源

任务

arn:aws:batch:*region*:*account*:job/*jobId*

条件键

`aws:ResourceTag/${TagKey}`（字符串）

根据与资源关联的标签筛选操作。

CreateComputeEnvironment

创建一个 AWS Batch 计算环境。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

条件键

aws:RequestTag/${TagKey}（字符串）

根据在请求中传递的标签筛选操作。

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

CreateJobQueue

创建一个 AWS Batch 作业队列。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

条件键

aws:RequestTag/${TagKey}（字符串）

根据在请求中传递的标签筛选操作。

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

DeleteComputeEnvironment

删除一个 AWS Batch 计算环境。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

CreateSchedulingPolicy

创建 AWS Batch 计划策略。

资源

计划策略

arn:aws:batch:##:##:计划-策略/##-##-##

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

条件键

aws:RequestTag/${TagKey}（字符串）

根据在请求中传递的标签筛选操作。

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

DeleteJobQueue

删除指定的作业队列。删除作业队列最终会删除队列中的所有作业。作业的删除速度约为每秒 16 个作业。

资源

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

DeleteSchedulingPolicy

删除指定的计划策略。

资源

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

DeregisterJobDefinition

取消注册 AWS Batch 作业定义。

资源

作业定义

arn:aws:batch:*region*:*account*:job-definition/*definition-name*:*revision*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

ListTagsForResource

列出指定资源的标签。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

任务

arn:aws:batch:*region*:*account*:job/*jobId*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业定义

arn:aws:batch:*region*:*account*:job-definition/*definition-name*:*revision*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

RegisterJobDefinition

注册一个 AWS Batch 定义。

资源

作业定义

arn:aws:batch:*region*:*account*:job-definition/*definition-name*

条件键

batch:AWSLogsCreateGroup（布尔值）

当此参数为 true 时，将会为日志创建 awslogs-group。

batch:AWSLogsGroup（字符串）

日志所在的 awslogs 组。

batch:AWSLogsRegion（字符串）

日志发送到的区域。

batch:AWSLogsStreamPrefix（字符串）

awslogs 日志流前缀。

batch:Image（字符串）

用于启动作业的 Docker 映像。

batch:LogDriver（字符串）

用于作业的日志驱动程序。

batch:Privileged（布尔值）

当此参数为 true 时，将会为该作业的容器提供对主机容器实例的提升权限。

batch:User（字符串）

要在该作业的容器中使用的用户名或数字 UID。

aws:RequestTag/${TagKey}（字符串）

根据在请求中传递的标签筛选操作。

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

SubmitJob

从作业定义提交 AWS Batch 作业。

资源

任务

arn:aws:batch:*region*:*account*:job/*jobId*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业定义

arn:aws:batch:*region*:*account*:job-definition/*definition-name*[:*revision*]

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

> **Note**
>
> 只有当作业定义Amazon 资源名称（ARN）的格式为 arn:aws:batch:*region*:*account_number*:job-definition/*definition-name*:*revision*. 时才能使用此密钥

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

TagResource

标记指定的资源。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

任务

arn:aws:batch:*region*:*account*:job/*jobId*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业定义

arn:aws:batch:*region*:*account*:job-definition/*definition-name*:*revision*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

条件键

aws:RequestTag/${TagKey}（字符串）

根据在请求中传递的标签筛选操作。

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

TerminateJob

终止 AWS Batch 作业队列中的作业。

资源

任务

arn:aws:batch:*region*:*account*:job/*jobId*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

UntagResource

取消对指定资源的标记。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

任务

arn:aws:batch:*region*:*account*:job/*jobId*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业定义

arn:aws:batch:*region*:*account*:job-definition/*definition-name*:*revision*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

条件键

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

UpdateComputeEnvironment

更新 AWS Batch 计算环境。

资源

计算环境

arn:aws:batch:*region*:*account*:compute-environment/*compute-environment-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

UpdateJobQueue

更新作业队列。

资源

作业队列

arn:aws:batch:*region*:*account*:job-queue/*queue-name*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

UpdateSchedulingPolicy

更新计划策略。

资源

计划策略

arn:aws:batch:*##*:*##*:计划-策略/*##-##-##*

条件键

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

AWS Batch API 操作的条件密钥

AWS Batch 定义以下在 IAM policy的 Condition 元素中使用的条件键。您可以使用这些键细化应用策略语句的条件。要查看对所有服务都可用的全局条件键，请参阅 IAM 用户指南中的可用的全局条件键。

batch:AWSLogsCreateGroup（布尔值）

当此参数为 true 时，将会为日志创建 awslogs-group。

batch:AWSLogsGroup（字符串）

日志所在的 awslogs 组。

batch:AWSLogsRegion（字符串）

日志发送到的 AWS 区域。

batch:AWSLogsStreamPrefix（字符串）

awslogs 日志流前缀。

batch:Image（字符串）

用于启动作业的 Docker 映像。

batch:LogDriver（字符串）

用于作业的日志驱动程序。

batch:Privileged（布尔值）

当此参数为 true 时，将对作业容器提供对主机容器实例的提升的权限（类似于根用户）。

aws:ResourceTag/${TagKey}（字符串）

根据与资源关联的标签筛选操作。

aws:RequestTag/${TagKey}（字符串）

根据在请求中传递的标签筛选操作。

batch:ShareIdentifier（字符串）

根据发送给 SubmitJob 的 shareIdentifier 参数筛选操作。

aws:TagKeys（字符串）

根据在请求中传递的标签键筛选操作。

batch:User（字符串）

要在容器中使用的用户名或数字用户 ID (uid)。

示例策略

以下示例显示了您可用于控制用户对于 AWS Batch 权限的策略语句。

示例

- 只读访问权限
- 示例：限制为 POSIX 用户、Docker 映像、权限级别和作业提交角色
- 示例：限于作业提交的作业定义前缀
- 限制作业队列
- 当条件所有键都匹配字符串时拒绝操作
- 当条件所有键都匹配字符串时拒绝操作
- 使用 batch:ShareIdentifier 条件键

只读访问权限

以下策略向用户授予使用名称以 `Describe` 和 `List` 开头的所有 AWS Batch API 操作的权限。

除非另一条语句授予权限，否则用户无权对资源执行任何操作。默认情况下，他们被拒绝使用 API 操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:Describe*",
                "batch:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

示例：限制为 POSIX 用户、Docker 映像、权限级别和作业提交角色

以下策略允许 POSIX 用户管理自己的一组受限作业定义。

第一个语句和第二个语句允许用户注册和取消注册其名称前缀为 *JobDefA_* 的任何作业定义。

第一个语句还使用条件上下文键来限制作业定义的 containerProperties 中的 POSIX 用户、特权状态和容器映像值。有关更多信息，请参阅 AWS Batch API 参考中的 RegisterJobDefinition。在此示例中，只有当 POSIX 用户设置为 nobody 时，才能注册作业定义。特权标志设置为 false。最后，在 Amazon ECR 存储库中将映像设置为 myImage。

> **Important**
>
> Docker 将 user 参数解析为该用户在容器映像中的 uid。在大多数情况下，可以在容器映像中的 /etc/passwd 文件中找到它。可以通过在作业定义和任何关联的IAM policy 中使用直接 uid 值来避免此名称解析。AWS Batch API 和 batch:User IAM 条件键都支持数字值。

第三个语句限制用户仅将特定角色传递给作业定义。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:RegisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/JobDefA_*"
            ],
            "Condition": {
                "StringEquals": {
                    "batch:User": [
                        "nobody"
                    ],
                    "batch:Image": [
                        "<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/myImage"
                    ]
                },
                "Bool": {
                    "batch:Privileged": "false"
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "batch:DeregisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/JobDefA_*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<aws_account_id>:role/MyBatchJobRole"
            ]
        }
    ]
}
```

示例：限于作业提交的作业定义前缀

以下策略允许用户将作业提交到任何作业队列，其任何作业定义名称以 *JobDefA_* 开头。

> **Important**
>
> 在限定作业提交的资源级访问时，必须同时提供作业队列和作业定义资源类型。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
      ],
      "Resource": [
        "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/JobDefA_*",
        "arn:aws:batch:<aws_region>:<aws_account_id>:job-queue/*"
      ]
    }
```

```
  ]
}
```

限制作业队列

以下策略允许用户将作业提交到名为 queue1 的特定作业队列，该队列具有任何作业定义名称。

> **Important**
>
> 在限定作业提交的资源级访问时，必须同时提供作业队列和作业定义资源类型。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
      ],
      "Resource": [
        "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/*",
        "arn:aws:batch:<aws_region>:<aws_account_id>:job-queue/queue1"
      ]
    }
  ]
}
```

当条件所有键都匹配字符串时拒绝操作

当 batch:Image（容器映像 ID）条件键为 *string1* 且 batch:LogDriver（容器日志驱动程序）条件键为 string *string2* 时，以下策略拒绝访问 RegisterJobDefinition API 操作。AWS Batch评估每个容器上的条件键。当作业跨越多个容器（例如多节点平行作业）时，容器可能会有不同的配置。如果在一个语句中评估多个条件键，则使用 AND 逻辑将它们组合在一起。因此，如果多个条件键中的任何一个与容器不匹配，则该 Deny 效果不会应用于该容器。相反，同一作业中的不同容器可能会被拒绝。

有关 AWS Batch 条件密钥的列表，请参阅《服务授权参考》中的 AWS Batch 的条件密钥。除 batch:ShareIdentifier 外，所有 batch 条件键都可以用这种方式使用。batch:ShareIdentifier 条件键是为作业定义的，而不是为作业定义定义的。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": "batch:RegisterJobDefinition",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "batch:Image": "string1",
          "batch:LogDriver": "string2"
        }
      }
    }
  ]
}
```

当条件所有键都匹配字符串时拒绝操作

当 batch:Image（容器映像 ID）条件键为 *string1* 或 batch:LogDriver（容器日志驱动程序）条件键为 *string string2* 时，以下策略拒绝访问 RegisterJobDefinition API 操作。当作业跨越多个容器（例如多节点平行作业）时，容器可能会有不同的配置。如果在一个语句中评估多个条件键，则使用 AND 逻辑将它们组合在一起。因此，如果多个条件键中的任何一个与容器不匹配，则该 Deny 效果不会应用于该容器。相反，同一作业中的不同容器可能会被拒绝。

有关 AWS Batch 条件密钥的列表，请参阅《服务授权参考》中的 AWS Batch 的条件密钥。除 batch:ShareIdentifier 外，所有 batch 条件键都可以用这种方式使用。（batch:ShareIdentifier 条件键是为作业定义的，而不是为作业定义定义的。）

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

```
      "Effect": "Allow",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "batch:Image": [
            "string1"
          ]
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "batch:RegisterJobDefinition"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "batch:LogDriver": [
            "string2"
          ]
        }
      }
    }
  ]
}
```

使用 batch:ShareIdentifier 条件键

使用以下策略将使用 jobDefA 作业定义的作业提交到具有 lowCpu 份额标识符的 jobqueue1 作业队列。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
      ],
      "Resource": [
        "arn:aws::batch:<aws_region>:<aws_account_id>:job-definition/JobDefA",
        "arn:aws::batch:<aws_region>:<aws_account_id>:job-queue/jobqueue1"
      ],
      "Condition": {
        "StringEquals": {
          "batch:ShareIdentifier": [
            "lowCpu"
          ]
        }
      }
    }
  ]
}
```

AWS Batch 托管式策略

AWS Batch 提供了一个托管策略，可将该策略附加到用户来提供使用 AWS Batch 资源和 API 操作的权限。您可以直接应用此策略，也可以以它为起点创建自己的策略。有关这些策略中提到的每个 API 操作的更多信息，请参阅 AWS Batch API 参考 中的 操作。

AWSBatchFullAccess

此策略授予对 AWS Batch 的完全管理员访问权限。

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

```
    {
      "Effect":"Allow",
      "Action":[
        "batch:*",
        "cloudwatch:GetMetricStatistics",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeVpcs",
        "ec2:DescribeImages",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribeLaunchTemplateVersions",
        "ecs:DescribeClusters",
        "ecs:Describe*",
        "ecs:List*",
        "eks:DescribeCluster",
        "eks:ListClusters",
        "logs:Describe*",
        "logs:Get*",
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "iam:ListInstanceProfiles",
        "iam:ListRoles"
      ],
      "Resource":"*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "iam:PassRole"
      ],
      "Resource":[
        "arn:aws:iam::*:role/AWSBatchServiceRole",
        "arn:aws:iam::*:role/service-role/AWSBatchServiceRole",
        "arn:aws:iam::*:role/ecsInstanceRole",
        "arn:aws:iam::*:instance-profile/ecsInstanceRole",
        "arn:aws:iam::*:role/iaws-ec2-spot-fleet-role",
        "arn:aws:iam::*:role/aws-ec2-spot-fleet-role",
        "arn:aws:iam::*:role/AWSBatchJobRole*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
```

```
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"arn:aws:iam::*:role/*Batch*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "batch.amazonaws.com"
          }
      }
    }
  ]
}
```

创建 AWS Batch IAM policy

可以创建特定的 IAM policy 来限制您账户中的用户有权访问的调用和资源。然后，您可以将这些策略附加到用户。

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户使用指定资源执行指定任务。有关更多信息，请参阅 IAM 用户指南中的权限与策略。有关如何管理和创建自定义 IAM policy 的说明，请参阅管理 IAM policy。

Amazon ECS 实例角色

AWS Batch 计算环境已填充有 Amazon ECS 容器实例。它们在本地运行 Amazon ECS 容器代理。Amazon ECS 容器代理将代表您调用各种 AWS API 操作。因此，运行该代理的容器实例需要一个 IAM policy 和角色，以便该服务了解该代理属于您。您必须创建一个 IAM 角色和一个实例配置文件，以便容器实例在启动时使用。否则，您无法创建计算环境并在其中启动容器实例。此要求适用于在使用或未使用由 Amazon 提供的经 Amazon ECS 优化的 AMI 的情况下启动的容器实例。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的容器实例 中的 Amazon ECS 容器实例 IAM 角色。

在控制台首次运行体验中将自动为您创建 Amazon ECS 实例角色和实例配置文件。但是，您可以按照以下步骤检查您的账户是否已有 Amazon ECS 实例角色和实例配置文件。以下步骤还介绍了如何附加托管 IAM policy。

在 IAM 控制台中检查 **ecsInstanceRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles（角色）。

3. 在角色列表中搜索 ecsInstanceRole。如果该角色不存在，请使用以下步骤创建该角色。

   a. 请选择 Create Role(创建角色)。

   b. 对于 Trusted entity type（可信实体类型），选择 AWS 服务。

   c. 对于 常用案例，选择 EC2。

   d. 选择 Next（下一步）。

   e. 要查看权限策略，请搜索 AmazonEC2ContainerServiceforEC2Role。

   f. 选中 AmazonEC2ContainerServiceforEC2Role 旁边的复选框，然后选择下一步。

   g. 对于 Role Name (角色名称)，键入 ecsInstanceRole，然后选择 Create Role (创建角色)。

   > **Note**
   >
   > 如果使用 AWS Management Console 创建 Amazon EC2 的角色，则控制台创建实例配置文件，将其命名为与角色相同的名称。

或者，您可以使用 AWS CLI 创建 ecsInstanceRole IAM 角色。使用以下示例通过信任策略和 AWS 托管策略，创建一个 IAM 角色。

创建 IAM 角色和实例配置文件（AWS CLI）

1. 创建以下信任策略并将其保存在名为 ecsInstanceRole-role-trust-policy.json 的文本文件中。

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

2. 使用 create-role 命令创建 ecsInstanceRole 角色。在 assume-role-policy-document 参数中指定信任策略文件的位置。

```
$ aws iam create-role \
    --role-name ecsInstanceRole \
    --assume-role-policy-document file://ecsInstanceRole-role-trust-policy.json
```

以下为响应示例。

```
{
    "Role": {
         "Path": "/",
         "RoleName: "ecsInstanceRole",
         "RoleId": "AROAT46P5RDIY4EXAMPLE",
         "Arn": "arn:aws:iam::123456789012:role/ecsInstanceRole".
        "CreateDate": "2022-12-12T23:46:37.247Z",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service: "ec2.amazonaws.com"
                    }
                    "Action": "sts:AssumeRole",
                 }
              ]
          }
        }
```

3. 使用 create-instance-profile 命令创建名为 `ecsInstanceRole` 的实例配置文件。

> **Note**
>
> 您需要在 AWS CLI 和 AWS API 中将角色和实例配置文件创建为单独的操作。

```
$ aws iam create-instance-profile --instance-profile-name ecsInstanceRole
```

以下为响应示例。

```
{
    "InstanceProfile": {
```

```
        "Path": "/",
        "InstanceProfileName": "ecsInstanceRole",
        "InstanceProfileId": "AIPAT46P5RDITREXAMPLE",
        "Arn": "arn:aws:iam::123456789012:instance-profile/ecsInstanceRole",
        "CreateDate": "2022-06-30T23:53:34.093Z",
        "Roles": [],    }
}
```

4. 使用 add-role-to-instance-profile 命令将 ecsInstanceRole 角色附加到 ecsInstanceRole 实例配置文件。

```
aws iam add-role-to-instance-profile \
    --role-name ecsInstanceRole --instance-profile-name ecsInstanceRole
```

5. 使用 attach-role-policy 命令将 AmazonEC2ContainerServiceforEC2RoleAWS托管策略附加到 ecsInstanceRole 角色。

```
$ aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role \
    --role-name ecsInstanceRole
```

Amazon EC2 竞价型实例集角色

如果创建一个使用 Amazon EC2 竞价型实例集实例的托管计算环境，则必须创建AmazonEC2SpotFleetTaggingRole策略。该政策授权竞价型实例集代表您启动、标记和终止实例。在竞价型实例集请求中指定该角色。还必须具有适用于 Amazon EC2 Spot 和竞价型实例集的 AWSServiceRoleForEC2Spot 和 AWSServiceRoleForEC2SpotFleet 服务相关角色。请按照以下说明以创建所有这些角色。有关更多信息，请参阅《IAM 用户指南》中的使用服务相关角色和创建将权限委托给AWS服务的角色。

主题

- 在AWS Management Console中创建 Amazon EC2 竞价型实例集角色
- 使用AWS CLI创建 Amazon EC2 竞价型实例集角色

在AWS Management Console中创建 Amazon EC2 竞价型实例集角色

要创建适用于 Amazon EC2 竞价型实例集的**AmazonEC2SpotFleetTaggingRole** IAM 服务相关角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在访问管理下，请选择角色。
3. 对于角色，选择创建角色。
4. 从为可信实体类型选择可信实体中，选择AWS 服务。
5. 对于其他AWS 服务用例，请选择EC2，然后选择 EC2 - 竞价型实例集标记。
6. 选择下一步。
7. 在策略名称的权限策略中，验证AmazonEC2SpotFleetTaggingRole。
8. 选择下一步。
9. 对于名称，请查看并创建：
   a. 对于命名标签，输入用于标识角色的名称。
   b. 在描述中，输入策略的简短解释。
   c. （可选）对于步骤 1：选择可信实体，选择编辑以修改代码。
   d. （可选）对于步骤 2：添加权限，选择编辑以修改代码。
   e. （可选）对于添加标签，选择添加标签以向资源添加标签。
   f. 选择创建角色。

> **Note**
>
> 过去，Amazon EC2 竞价型实例集角色有两个托管策略。
>
> - AmazonEC2SpotFleetRole：这是竞价型实例集角色的初始托管策略。但是，不再建议将其与AWS Batch一起使用。此策略不支持计算环境中的竞价型实例标记，这是使用AWSServiceRoleForBatch服务相关角色所必需的。如果以前是使用此策略创建的竞价型实例集角色，请将新的推荐策略应用于该角色。有关更多信息，请参阅创建时未标记的竞价型实例。
> - AmazonEC2SpotFleetTaggingRole：该角色提供标记 Amazon EC2 竞价型实例的所需的所有权限。使用此角色允许在AWS Batch计算环境中标记竞价型实例。

使用AWS CLI创建 Amazon EC2 竞价型实例集角色

要为竞价型实例集计算环境创建 AmazonEC2SpotFleetTaggingRole IAM 角色

1. 使用AWS CLI运行以下命令。

```
$ aws iam create-role --role-name AmazonEC2SpotFleetTaggingRole \
    --assume-role-policy-document '{
    "Version":"2012-10-17",
    "Statement":[
       {
         "Sid":"",
         "Effect":"Allow",
         "Principal": {
           "Service":"spotfleet.amazonaws.com"
          },
          "Action":"sts:AssumeRole"
        }
      ]
    }'
```

2. 要将 AmazonEC2SpotFleetTaggingRole 托管 IAM policy 附加到AmazonEC2SpotFleetTaggingRole 角色上，请使用AWS CLI运行以下命令。

```
$ aws iam attach-role-policy \
  --policy-arn \
    arn:aws:iam::aws:policy/service-role/AmazonEC2SpotFleetTaggingRole \
  --role-name \
    AmazonEC2SpotFleetTaggingRole
```

创建适用于 Amazon EC2 Spot 的 **AWSServiceRoleForEC2Spot** IAM 服务相关角色

> **Note**
>
> 如果AWSServiceRoleForEC2Spot IAM 服务相关角色已存在，则会出现类似于以下内容的错误消息。
>
> ```
> An error occurred (InvalidInput) when calling the CreateServiceLinkedRole
>  operation:
> ```

```
Service role name AWSServiceRoleForEC2Spot has been taken in this account,
 please try a different suffix.
```

- 使用AWS CLI运行以下命令。

```
$ aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
```

要创建适用于 Amazon EC2 竞价型实例集的**AWSServiceRoleForEC2SpotFleet** IAM 服务相关角色

> **Note**
>
> 如果AWSServiceRoleForEC2SpotFleet IAM 服务相关角色已存在，则会出现类似于以下内容的错误消息。
>
> ```
> An error occurred (InvalidInput) when calling the CreateServiceLinkedRole
>  operation:
> Service role name AWSServiceRoleForEC2SpotFleet has been taken in this account,
>  please try a different suffix.
> ```

- 使用AWS CLI运行以下命令。

```
$ aws iam create-service-linked-role --aws-service-name spotfleet.amazonaws.com
```

EventBridge IAM 角色

Amazon EventBridge 提供近乎实时的系统事件流，这些系统事件描述 AWS资源的变化。AWS Batch作业可作为 EventBridge 目标提供。通过使用可快速设置的简单规则，可以匹配事件并提交AWS Batch任务以响应这些事件。EventBridge 必须获得代表您运行AWS Batch作业的权限，然后才能提交包含 EventBridge 规则和目标的AWS Batch作业。

> **Note**
>
> 在 EventBridge 控制台中创建将AWS Batch队列指定为目标的规则时，您可以创建此角色。有关示例演练的信息，请参阅AWS Batch 以就业为 EventBridge 目标。可以使用 IAM 控制台手动创建 EventBridge 角色。有关说明，请参阅《IAM 用户指南》中的使用自定义信任策略创建角色（控制台）。

EventBridge IAM 角色的信任关系必须为`events.amazonaws.com`服务主体提供代入该角色的能力。

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

请确保附加到 EventBridge IAM 角色的策略`batch:SubmitJob`允许访问资源。在以下示例中，AWS Batch给出了提供这些权限的`AWSBatchServiceEventTargetRole`托管策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
       ],
      "Resource": "*"
    }
  ]
}
```

AWS Batch Amazon 活动直播 EventBridge

您可以使用适用于 Amazon AWS Batch 的事件流 EventBridge 来接收有关任务队列中任务当前状态的近乎实时的通知。

您可以使用 EventBridge 来进一步了解您的 AWS Batch 服务。更具体地说，您可以使用它来检查作业进度、构建 AWS Batch 自定义工作流程、生成使用情况报告或指标，或者构建自己的仪表板。使用 AWS Batch 和 EventBridge，您无需使用可以持续轮询 AWS Batch 作业状态变化的调度和监控代码。相反，您可以使用各种 Amazon EventBridge 目标异步处理 AWS Batch 任务状态更改。其中包括亚马逊简单队列服务 AWS Lambda、亚马逊简单通知服务或亚马逊 Kinesis Data Streams。

确保 AWS Batch 事件流中的事件至少传送一次。在发送重复事件的情况下，事件会提供足量信息来确定重复项。这样就可以比较事件的时间戳和作业状态。

AWS Batch 可以将工作作为 EventBridge 目标。使用简单的规则，您可以匹配事件并根据事件提交 AWS Batch 作业。有关更多信息，请参阅什么是 EventBridge？ 在《亚马逊 EventBridge 用户指南》中。您还可以使用cron或评分表达式 EventBridge 来安排在特定时间自动触发的自动操作。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的创建按计划运行的亚马逊 EventBridge 规则。有关示例演练的信息，请参阅AWS Batch 以就业为 EventBridge 目标。有关使用 EventBridge 计划程序的信息，请参阅《亚马逊 EventBridge 用户指南》中的 “设置亚马逊 EventBridge 日程安排”。

主题

- AWS Batch 活动
- 将 AWS 用户通知与 AWS Batch
- AWS Batch 以就业为 EventBridge 目标
- 教程：侦听 AWS Batch EventBridge
- 教程：针对作业失败事件发送 Amazon Simple Notification Service 警报

AWS Batch 活动

AWS Batch 向。发送任务状态更改事件 EventBridge。 AWS Batch 跟踪您的工作状态。如果先前提交的作业的状态发生变化，则会调用一个事件。例如，如果状态为`RUNNING`的作业变为`FAILED`状态。这些事件归类为作业状态更改事件。

> **Note**
>
> AWS Batch 将来可能会添加其他事件类型、来源和细节。如果以编程方式对事件 JSON 数据反序列化，请确保应用程序已准备好处理未知属性。这是为了避免在添加这些附加属性时出现问题。

作业状态更改事件

只要现有 (以前提交的) 作业状态发生更改，就会创建一个事件。有关 AWS Batch 作业状态的更多信息，请参阅作业状态。

> **Note**
>
> 对于初始作业提交不会创建事件。

Example 作业状态更改事件

作业状态更改事件以下面的形式传送。该`detail`部分类似于《API 参考》中从 DescribeJobsAPI 操作返回的JobDetailAWS Batch 对象。有关 EventBridge 参数的更多信息，请参阅 Amazon EventBridge 用户指南中的事件和事件模式。

```
{
    "version": "0",
    "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
    "detail-type": "Batch Job State Change",
    "source": "aws.batch",
    "account": "123456789012",
    "time": "2022-01-11T23:36:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-ba5a-4727fcce14a8"
    ],
    "detail": {
        "jobArn": "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
        "jobName": "event-test",
        "jobId": "4c7599ae-0a82-49aa-ba5a-4727fcce14a8",
        "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
PexjEHappyPathCanary2JobQueue",
```

```
        "status": "RUNNABLE",
        "attempts": [],
        "createdAt": 1641944200058,
        "retryStrategy": {
            "attempts": 2,
            "evaluateOnExit": []
        },
        "dependsOn": [],
        "jobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/first-
run-job-definition:1",
        "parameters": {},
        "container": {
            "image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest",
            "command": [
                "sleep",
                "600"
            ],
            "volumes": [],
            "environment": [],
            "mountPoints": [],
            "ulimits": [],
            "networkInterfaces": [],
            "resourceRequirements": [
                {
                    "value": "2",
                    "type": "VCPU"
                }, {
                    "value": "256",
                    "type": "MEMORY"
                }
            ],
            "secrets": []
        },
        "tags": {
            "resourceArn": "arn:aws:batch:us-
east-1:123456789012:job/4c7599ae-0a82-49aa-ba5a-4727fcce14a8"
        },
        "propagateTags": false,
        "platformCapabilities": []
    }
}
```

Job 队列被屏蔽的事件

每当 AWS Batch 检测到处于该RUNNABLE状态的任务从而阻塞队列时，都会在 Amazon Events 中创建一个 CloudWatch 事件。有关支持的队列阻塞原因的更多信息，请参阅任务队列消息被阻塞的示例。DescribeJobsAPI 操作的statusReason字段中也有同样的原因。

Example 作业状态更改事件

作业状态更改事件以下面的形式传送。该detail部分类似于《API 参考》中从 DescribeJobsAPI 操作返回的JobDetailAWS Batch 对象。有关 EventBridge参数的更多信息，请参阅 Amazon EventBridge 用户指南中的事件和事件模式。

```
{
    "version": "0",
    "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
    "detail-type": "Batch Job Queue Blocked",
    "source": "aws.batch",
    "account": "123456789012",
    "time": "2022-01-11T23:36:40Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
        "arn:aws:batch:us-east-1:123456789012:job-queue/PexjEHappyPathCanary2JobQueue"
    ],
    "detail": {
        "jobArn": "arn:aws:batch:us-east-1:123456789012:job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
        "jobName": "event-test",
        "jobId": "4c7599ae-0a82-49aa-ba5a-4727fcce14a8",
        "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
PexjEHappyPathCanary2JobQueue",
        "status": "RUNNABLE",
        "statusReason": "blocked-reason"
        "attempts": [],
        "createdAt": 1641944200058,
        "retryStrategy": {
            "attempts": 2,
            "evaluateOnExit": []
        },
        "dependsOn": [],
        "jobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/first-
run-job-definition:1",
```

```
        "parameters": {},
        "container": {
            "image": "137112412989.dkr.ecr.us-east-1.amazonaws.com/amazonlinux:latest",
            "command": [
                "sleep",
                "600"
            ],
            "volumes": [],
            "environment": [],
            "mountPoints": [],
            "ulimits": [],
            "networkInterfaces": [],
            "resourceRequirements": [
                {
                    "value": "2",
                    "type": "VCPU"
                }, {
                    "value": "256",
                    "type": "MEMORY"
                }
            ],
            "secrets": []
        },
        "tags": {
            "resourceArn": "arn:aws:batch:us-
east-1:123456789012:job/4c7599ae-0a82-49aa-ba5a-4727fcce14a8"
        },
        "propagateTags": false,
        "platformCapabilities": []
    }
}
```

将 AWS 用户通知与 AWS Batch

可以使用 AWS 用户通知来设置交付渠道，以获得有关 AWS Batch 事件的通知。当事件与指定的规则匹配时，会收到通知。可以通过多个渠道接收事件通知，包括电子邮件、AWS Chatbot聊天通知或AWS Console Mobile Application推送通知。还可以在控制台通知中心中查看通知。用户通知支持聚合，这可以减少在具体事件期间收到的通知数量。

要在中配置用户通知，请执行 AWS Batch以下操作：

1. 打开AWS Batch 控制台。

2. 选择控制面板。
3. 选择配置通知。
4. 在 AWS 用户通知中，选择创建通知配置。

有关如何配置和查看用户通知的更多信息，请参阅AWS 用户通知入门。

AWS Batch 以就业为 EventBridge 目标

亚马逊 EventBridge提供了近乎实时的系统事件流，这些事件描述了亚马逊 Web Services 资源的变化。通常， AWS Batch 在亚马逊弹性容器服务上，亚马逊 Elastic Kubernetes Service AWS 和 Fargate 作业可作为目标使用。 EventBridge 使用简单的规则，您可以匹配事件并根据事件提交 AWS Batch 作业。有关更多信息，请参阅什么是 EventBridge？ 在《亚马逊 EventBridge 用户指南》中。

您还可以使用cron或评分表达式 EventBridge 来安排在特定时间调用的自动操作。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的创建按计划运行的亚马逊 EventBridge 规则。

有关如何创建在事件与事件模式匹配时运行的规则的信息，请参阅 EventBridge 《亚马逊 EventBridge 用户指南》中的创建对事件做出反应的 Amazon 规则。

以 AWS Batch 作业为 EventBridge 目标的常见用例包括以下用例：

- 计划的作业以固定的时间间隔出现。例如，只有在 Amazon EC2 竞价型实例价格较低时，cron 作业才会在使用率低的时段出现。
- AWS Batch 作业是为了响应已登录的 API 操作而运行的 CloudTrail。例如，只要将对象上传到指定的 Amazon S3 存储桶，就会提交作业。每次发生这种情况时， EventBridge 输入转换器都会将对象的存储桶和密钥名称传递给 AWS Batch 参数。

> **Note**
>
> 在这种情况下，所有相关 AWS 资源都必须位于同一个区域。这包括 Amazon S3 存储桶、EventBridge 规则和 CloudTrail 日志等资源。

在提交带有 EventBridge 规则和目标的 AWS Batch 作业之前，该 EventBridge 服务需要多个权限才能运行 AWS Batch 作业。在 EventBridge 控制台中创建将 AWS Batch 任务指定为目标的规则时，也可以创建此角色。有关此角色所需的服务委托人和 IAM 权限的更多信息，请参阅 EventBridge IAM 角色。

创建计划 AWS Batch 作业

以下过程介绍如何创建计划 AWS Batch 任务和所需的 EventBridge IAM 角色。

使用创建计划 AWS Batch 作业 EventBridge

> **Note**
>
> 此过程适用于所有亚马逊 ECS、Amaz AWS Batch on EKS 和 AWS Fargate 任务。

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择规则。
4. 选择创建规则。
5. 对于名称，为计算环境指定唯一名称。名称最多可以包含 64 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

   > **Note**
   >
   > 规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。

6. （可选）对于描述，输入规则的描述。
7. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择默认。当你账户 AWS 服务 中的某人发出事件时，它总是会转到你账户的默认事件总线。
8. （可选）如果您不想立即运行所选总线上的规则，请关闭该规则。
9. 对于 Rule type（规则类型），选择 Schedule（计划）。
10. 选择继续创建规则，或者选择下一步。
11. 对于 Schedule pattern（计划模式），执行以下操作之一：

    - 选择在特定时间（例如上午 8:00）运行的精细计划。每月第一个星期一，太平洋标准时间，然后输入 cron 表达式。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Cron 表达式。
    - 选择以常规速率运行的计划，例如每 10 分钟，然后输入 rate 表达式。

12. 选择下一步。

13. 对于 Target types（目标类型），选择 AWS 服务。
14. 在选择目标中，选择批处理作业队列。然后，进行以下配置：
    - Job queue（作业队列）：输入您在其中计划作业的作业队列的 Amazon 资源名称（ARN）。
    - Job definition (任务定义)：输入要用于任务的任务定义的名称和版本或完整 ARN。
    - Job name (任务名称)：输入您的任务的名称。
    - Array size (数组大小)：(可选) 输入要运行多个副本的任务的数组大小。有关更多信息，请参阅数组作业。
    - Job attempts (任务尝试次数)：(可选) 输入任务失败时重试的次数。有关更多信息，请参阅 自动作业重试。
15. 对于 Batch 作业队列目标类型， EventBridge 需要向目标发送事件的权限。 EventBridge 可以创建规则运行所需的 IAM 角色。请执行以下操作之一：
    - 要自动创建 IAM 角色，请选择为此特定资源创建新角色。
    - 要使用您已经创建的 IAM 角色，请选择 使用现有角色。
16. （可选）展开 Additional settings (其他设置)。
    a. 在配置目标输入中，选择如何处理事件中的文本，然后再将其传递到目标。
    b. 对于事件的最大期限，请指定未处理事件保留多长时间的时间间隔。
    c. 对于重试次数，请输入事件的重试次数。
    d. 对于死信队列，选择一个选项来说明如何处理未处理的事件。如有必要，指定要用作死信队列的 Amazon SQS 队列。
17. （可选）选择 Add another target（添加其他目标），以为此规则添加其他目标。
18. 选择下一步。
19. （可选）在标签中，选择添加新标签以为规则添加资源标签。有关更多信息，请参阅 Amazon EventBridge 标签。
20. 选择下一步。
21. 对于查看和创建，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择 Create（创建）。

有关创建规则的更多信息，请参阅《亚马逊 EventBridge 用户指南》中的创建按计划运行的亚马逊 EventBridge 规则。

创建具有事件模式的规则

以下过程介绍如何使用事件模式创建规则。

创建在事件与定义的模式匹配时将事件发送到目标的规则

> **Note**
>
> 此过程适用于所有亚马逊 ECS、Amaz AWS Batch on EKS 和 AWS Fargate 任务。

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择规则。
4. 选择创建规则。
5. 对于名称，为计算环境指定唯一名称。名称最多可以包含 64 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

   > **Note**
   >
   > 规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。

6. （可选）对于描述，输入规则的描述。
7. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择默认。当你账户 AWS 服务 中的某人发出事件时，它总是会转到你账户的默认事件总线。
8. （可选）如果您不想立即运行所选总线上的规则，请关闭该规则。
9. 对于规则类型，选择具有事件模式的规则。
10. 选择下一步。
11. 在 “事件来源” 中，选择AWS 事件或 EventBridge 合作伙伴事件。
12. （可选）对于示例事件：

    a. 对于示例事件类型，选择AWS 事件。

    b. 对于示例事件，选择批处理作业状态更改。

13. 对于创建方法，选择使用模式表单。
14. 对于事件模式：

a. 对于事件源，选择 AWS 服务。
b. 对于 AWS 服务，选择批处理。
c. 对于事件类型，选择批量作业状态更改。

15. 选择下一步。
16. 对于 Target types（目标类型），选择 AWS 服务。
17. 在选择目标中，选择目标类型。例如，选择批处理作业队列。然后指定以下内容：

- Job queue（作业队列）：输入您在其中计划作业的作业队列的 Amazon 资源名称（ARN）。
- Job definition (任务定义)：输入要用于任务的任务定义的名称和版本或完整 ARN。
- Job name (任务名称)：输入您的任务的名称。
- Array size (数组大小)：(可选) 输入要运行多个副本的任务的数组大小。有关更多信息，请参阅数组作业。
- Job attempts (任务尝试次数)：(可选) 输入任务失败时重试的次数。有关更多信息，请参阅 自动作业重试。

18. 对于 Batch 作业队列目标类型， EventBridge 需要向目标发送事件的权限。 EventBridge 可以创建规则运行所需的 IAM 角色。请执行以下操作之一：

- 要自动创建 IAM 角色，请选择为此特定资源创建新角色。
- 要使用您之前创建的 IAM 角色，请选择使用现有角色。

19. （可选）展开 Additional settings (其他设置)。

a. 在配置目标输入中，选择如何处理事件中的文本。
b. 对于事件的最大期限，请指定未处理事件保留多长时间的时间间隔。
c. 对于重试次数，请输入事件的重试次数。
d. 对于死信队列，选择一个选项来说明如何处理未处理的事件。如有必要，指定要用作死信队列的 Amazon SQS 队列。

20. （可选）选择 添加其他目标，以添加其他目标。
21. 选择下一步。
22. （可选）在标签中，选择添加新标签以添加资源标签。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge 标签。
23. 选择下一步。
24. 对于查看和创建，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择 创建规则。

有关创建规则的更多信息，请参阅《亚马逊 EventBridge 用户指南》中的创建对事件做出反应的 Amazon EventBridge 规则。

使用 EventBridge 输入变压器按计划将事件信息传递给 T AWS Batch arget

在作业提交中，您可以使用 EventBridge 输入转换器将 AWS Batch 事件信息传递给。如果您因其他 AWS 事件信息而调用作业，则这可能特别有用。例如，将文件元上载到 Amazon S3 存储桶。您还可以在容器的命令中使用带有参数替换值的作业定义。 EventBridge 输入变压器可以根据事件数据提供参数值。

然后，您创建一个 AWS Batch 事件目标，该目标解析启动它的事件中的信息并将其转换为对象。`parameters`运行作业时，触发事件中的参数将传递至作业容器的命令。

> **Note**
>
> 在这种情况下，所有 AWS 资源（例如 Amazon S3 存储桶、 EventBridge 规则和 CloudTrail日志）必须位于同一个区域。

创建使用输入变压器的 AWS Batch 目标

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航栏中，选择 AWS 区域 要使用的。
3. 在导航窗格中，选择规则。
4. 选择创建规则。
5. 对于名称，为计算环境指定唯一名称。名称最多可以包含 64 个字符。可以包含大小写字母、数字、连字符（-）和下划线（_）。

   > **Note**
   >
   > 一条规则不能与同一事件总线上的另一条规则同名。 AWS 区域
6. （可选）对于描述，输入规则的描述。
7. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择默认。当你账户 AWS 服务 中的某人发出事件时，它总是会转到你账户的默认事件总线。

8. （可选）如果您不想立即运行所选总线上的规则，请关闭该规则。
9. 对于 Rule type（规则类型），选择 Schedule（计划）。
10. 选择继续创建规则，或者选择下一步。
11. 对于 Schedule pattern（计划模式），执行以下操作之一：
    - 选择在特定时间（例如上午 8:00）运行的精细计划。每月第一个星期一，太平洋标准时间，然后输入 cron 表达式。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Cron 表达式。
    - 选择以常规速率运行的计划，例如每 10 分钟，然后输入 rate 表达式。
12. 选择下一步。
13. 对于 Target types（目标类型），选择 AWS 服务。
14. 在选择目标中，选择批处理作业队列。然后，进行以下配置：
    - Job queue（作业队列）：输入您在其中计划作业的作业队列的 Amazon 资源名称（ARN）。
    - Job definition (任务定义)：输入要用于任务的任务定义的名称和版本或完整 ARN。
    - Job name (任务名称)：输入您的任务的名称。
    - Array size (数组大小)：(可选) 输入要运行多个副本的任务的数组大小。有关更多信息，请参阅数组作业。
    - Job attempts (任务尝试次数)：(可选) 输入任务失败时重试的次数。有关更多信息，请参阅 自动作业重试。
15. 对于 Batch 作业队列目标类型， EventBridge 需要向目标发送事件的权限。 EventBridge 可以创建规则运行所需的 IAM 角色。请执行以下操作之一：
    - 要自动创建 IAM 角色，请选择为此特定资源创建新角色。
    - 要使用您已经创建的 IAM 角色，请选择 使用现有角色。
16. （可选）展开 Additional settings (其他设置)。
17. 在 Additional settings（其他设置）部分，对于 Configure target input（配置目标输入），请选择 Input Transformer（输入转换器）。
18. 选择 Configure input transformer（配置输入转换器）。
19. （可选）对于示例事件：
    a. 对于示例事件类型，选择AWS 事件。
    b. 对于示例事件，选择批处理作业状态更改。

20. 在 Target input transformer（目标输入转换器）部分，对于 Input path（输入路径），请指定要从触发事件中解析的值。例如，要解析批处理作业状态更改事件，请使用以下 JSON 格式。

```
{
    "instance": "$.detail.jobId",
    "state": "$.detail.status"
}
```

21. 对于模板正文，输入以下模板：

```
{
    "instance": <jobId> ,
    "status": <status>
}
```

22. 选择确认。
23. 对于事件的最大期限，请指定未处理事件保留多长时间的时间间隔。
24. 对于重试次数，请输入事件的重试次数。
25. 对于死信队列，选择一个选项来说明如何处理未处理的事件。如有必要，指定要用作死信队列的 Amazon SQS 队列。
26. （可选）选择 添加其他目标，以添加其他目标。
27. 选择下一步。
28. （可选）在标签中，选择添加新标签以添加资源标签。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge 标签。
29. 选择下一步。
30. 对于查看和创建，请查看配置步骤。如果需要进行更改，请选择 Edit（编辑）。完成后，选择 创建规则。

教程：侦听 AWS Batch EventBridge

在本教程中，您设置一个简单的 AWS Lambda 函数，该函数会侦听 AWS Batch 任务事件并将这些事件写出到 CloudWatch Logs 日志流。

先决条件

本教程假定您具备可正常工作的计算环境和作业队列，随时可以接受作业。如果您没有要从中捕获事件的正在运行的计算环境和作业队列，请执行入门 AWS Batch中的步骤创建一个。在本教程结束时，您可选择向此作业队列提交作业，以测试您是否已正确配置 Lambda 函数。

步骤 1：创建 Lambda 函数

在此过程中，您将创建一个简单的 Lambda 函数来充当 AWS Batch 事件流消息的目标。

创建目标 Lambda 函数

1. 打开 AWS Lambda 控制台，地址：https://console.aws.amazon.com/lambda/。
2. 选择 Create function（创建函数），然后选择 Author from scratch（从头开始创作）。
3. 对于函数名称，请输入 batch-event-stream-handler。
4. 对于运行时系统，选择 Python 3.8。
5. 选择 Create function（创建函数）。
6. 在代码源部分中，编辑示例代码以匹配以下示例：

```
import json


def lambda_handler(event, _context):
    # _context is not used
    del _context
    if event["source"] != "aws.batch":
        raise ValueError("Function only supports input from events with a source
 type of: aws.batch")

    print(json.dumps(event))
```

   这是一个简单的 Python 3.8 函数，可输出由 AWS Batch 发送的事件。如果一切配置正确，则在本教程结束时，您将在与此 Lambda 函数关联的 CloudWatch Logs 日志流中看到事件详细信息。
7. 选择 Deploy（部署）。

步骤 2：注册事件规则

在该部分中，您创建一个 EventBridge 事件规则，用于捕获来自 AWS Batch 资源的任务事件。该规则捕获来自定义该规则的账户中的 AWS Batch 的所有事件。作业消息本身包含有关事件源的信息 (包括将事件源提交到其中的作业队列)。您可以使用此信息以编程方式过滤和排序事件。

> **Note**
>
> 在使用 AWS Management Console 创建事件规则时，控制台会自动为 EventBridge 添加 IAM 权限以调用 Lambda 函数。但是，如果您使用 AWS CLI 创建事件规则，则必须明确授予权限。有关更多信息，请参阅《Amazon EventBridge 用户指南》中的事件和事件模式。

创建 EventBridge 规则

1. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
2. 在导航窗格中，选择 Rules (规则)。
3. 选择 Create rule (创建规则)。
4. 为规则输入名称和描述。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于 Event bus（事件总线），请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 默认事件总线。当您账户中的某个 AWS 服务发出一个事件时，它始终会发送到您账户的默认事件总线。
6. 对于 Rule type（规则类型），选择 Rule with an event pattern（具有事件模式的规则）。
7. 选择 Next（下一步）。
8. 对于 Event source（事件源），选择 Other（其他）。
9. 对于事件模式，选择自定义模式（JSON 编辑器）。
10. 在文本区域中粘贴以下事件模式。

```
{
  "source": [
    "aws.batch"
  ]
}
```

此规则适用于您的所有 AWS Batch 组和每个 AWS Batch 事件。或者，您也可以创建一个更具体的规则来过滤掉一些结果。

11. 选择 Next（下一步）。
12. 对于 Target types（目标类型），选择 AWS service（服务）。
13. 对于选择目标，请选择 Lambda 函数，然后选择您的 Lambda 函数。
14. （可选）对于 Additional settings（其他设置），执行以下操作：

    a. 对于 Maximum age of event（事件的最大时长），输入一分钟（00:01）与 24 小时（24:00）之间的值。

    b. 对于重试尝试，输入 0 到 185 之间的数字。

    c. 对于死信队列，选择是否使用标准 Amazon SQS 队列作为死信队列。如果与此规则匹配的事件未成功传递到目标，EventBridge 会将这些事件发送到死信队列。请执行下列操作之一：

    - 选择无不使用死信队列。
    - 选择在当前 AWS 账户中选择一个 Amazon SQS 队列用作死信队列，然后从下拉列表中选择要使用的队列。
    - 选择在其他 Amazon SQS 队列中选择其他队列 AWS 帐户作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予 EventBridge 向其发送消息的权限。有关更多信息，请参阅 Amazon EventBridge 用户指南中的授予死信队列的权限。

15. 选择 Next（下一步）。
16. （可选）为规则输入一个或多个标签。有关更多信息，请参阅 Amazon EventBridge 用户指南中的 Amazon EventBridge 标签。
17. 选择 Next（下一步）。
18. 查看规则详细信息并选择 Create rule（创建规则）。

第 3 步：测试您的配置

您现在可以通过向作业队列提交作业来测试您的 EventBridge 配置。如果所有配置都正确完成，您的 Lambda 函数将触发并将事件数据写入到该函数的 CloudWatch Logs 日志流。

测试配置

1. 打开 AWS Batch 控制台，地址：https://console.aws.amazon.com/batch/。
2. 提交新的 AWS Batch 作业。有关更多信息，请参阅提交作业。

3. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
4. 在导航窗格中，选择 Logs (日志)，然后选择 Lambda 函数的日志组（例如，/aws/lambda/*my-function*）。
5. 选择日志流以查看事件数据。

教程：针对作业失败事件发送 Amazon Simple Notification Service 警报

在本教程中，您将配置一条 EventBridge 事件规则，该规则仅捕获作业已变为FAILED状态的作业事件。在本教程结束时，您还可以选择向该作业队列提交作业。这是为了测试您是否正确配置了您的 Amazon SNS 提醒。

先决条件

本教程假定您具备可正常工作的计算环境和作业队列，随时可以接受作业。如果您没有要从中捕获事件的正在运行的计算环境和作业队列，请执行入门 AWS Batch中的步骤创建一个。

步骤 1：创建并订阅 Amazon SNS 主题

在本教程中，您配置一个 Amazon SNS 主题来充当新事件规则的事件目标。

创建 Amazon SNS 主题

1. 通过 https://console.aws.amazon.com/sns/v3/home 打开 Amazon SNS 控制台。
2. 依次选择 Topics (主题) 和 Create topic (创建主题)。
3. 对于类型，选择标准。
4. 对于名称，输入 **`JobFailedAlert`** 并选择创建主题。
5. 在JobFailedAlert屏幕上，选择创建订阅。
6. 对于协议，选择电子邮件。
7. 对于端点，输入您当前有权访问的电子邮件地址，然后选择 创建订阅。
8. 检查您的电子邮件账户，并等待接收订阅确认电子邮件。在收到此电子邮件后，选择 确认订阅。

步骤 2：注册事件规则

接下来，注册一个仅捕获作业失败事件的事件规则。

注册您的 EventBridge 规则

1. 打开亚马逊 EventBridge 控制台，网址为 https://console.aws.amazon.com/events/。
2. 在导航窗格中，选择规则。
3. 选择创建规则。
4. 为规则输入名称和描述。

   规则不能与同一区域中的另一个规则和同一事件总线上的名称相同。
5. 对于事件总线，请选择要与此规则关联的事件总线。如果您希望此规则对来自您自己的账户的匹配事件触发，请选择 AWS 默认事件总线。当你账户中的某项 AWS 服务发出事件时，它总是会进入你账户的默认事件总线。
6. 对于规则类型，选择具有事件模式的规则。
7. 选择下一步。
8. 对于事件源，选择其他。
9. 对于事件模式，选择 自定义模式（JSON 编辑器）。
10. 在文本区域中粘贴以下事件模式。

```
{
  "detail-type": [
    "Batch Job State Change"
  ],
  "source": [
    "aws.batch"
  ],
  "detail": {
    "status": [
      "FAILED"
    ]
  }
}
```

   此代码定义了一 EventBridge 条规则，该规则与任务状态为的任何事件相匹配`FAILED`。有关事件模式的更多信息，请参阅 Amazon EventBridge 用户指南中的事件和事件模式。
11. 选择 Next（下一步）。
12. 对于目标类型，选择AWS 服务。
13. 在 “选择目标” 中，选择 SNS 主题，在 “主题” 中选择JobFailedAlert。
14. （可选）对于 Additional settings（其他设置），执行以下操作：

a. 对于 Maximum age of event（事件的最大时长），输入一分钟（00:01）与 24 小时（24:00）之间的值。

b. 对于重试尝试，输入 0 到 185 之间的数字。

c. 对于死信队列，选择是否使用标准的 Amazon SQS 队列作为死信队列。 EventBridge 如果匹配此规则的事件未成功传送到目标，则将其发送到死信队列。请执行以下操作之一：

- 选择无不使用死信队列。
- 在当前 AWS 账户中选择要用作死信队列的 Amazon SQS 队列，然后从下拉列表中选择要使用的队列。
- 选择选择其他 AWS 账户中的 Amazon SQS 队列作为死信队列，然后输入要使用的队列的 ARN。您必须将基于资源的策略附加到队列，以授予向该队列发送消息的 EventBridge 权限。有关更多信息，请参阅 A mazon EventBridge 用户指南中的向死信队列授予权限。

15. 选择 Next（下一步）。
16. （可选）为规则输入一个或多个标签。有关更多信息，请参阅《亚马逊 EventBridge 用户指南》中的亚马逊 EventBridge 标签。
17. 选择下一步。
18. 查看规则详细信息并选择创建规则。

步骤 3：测试您的规则

要测试您的规则，请提交一个在启动后很快就以非零退出代码退出的作业。如果您的事件规则配置正确，您将在几分钟内收到包含事件文本的电子邮件消息。

测试规则

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 提交一份新 AWS Batch 工作。有关更多信息，请参阅 提交作业。对于作业的命令，替换为以下命令，以退出代码 1 退出容器。

```
/bin/sh, -c, 'exit 1'
```

3. 查看您的电子邮件以确认您已收到失败作业通知的电子邮件提醒。

替代规则：Batch Job 队列已阻止

要创建监控 B atch Job Queue Bloc ked 的事件规则，请重复本教程中的步骤，并进行以下更改：

1. 在步骤 1 中，使用*BlockedJobQueue*作为主题名称。
2. 在步骤 2 中，在 JSON 编辑器中使用以下模式：

```
{
   "detail-type": [
     "Batch Job Queue Blocked"
   ],
   "source": [
     "aws.batch"
   ]
}
```

将 CloudWatch 日志与配合使用 AWS Batch

您可以在 EC2 资源上配置 AWS Batch 任务，将详细的日志信息和指标发送到 CloudWatch 日志。这样就可以在同一个方便的位置查看作业中的不同日志。有关 CloudWatch 日志的更多信息，请参阅什么是 Amazon CloudWatch 日志？ 在《亚马逊 CloudWatch 用户指南》中。

> **Note**
>
> 默认情况下， AWS Fargate 容器的 CloudWatch 日志处于启用状态。

要开启和自定义 CloudWatch 日志记录，请查看以下一次性配置任务：

- 对于基于 EC2 资源的 AWS Batch 计算环境，请向`ecsInstanceRole`角色添加 IAM 策略。有关更多信息，请参阅 the section called “添加 CloudWatch 日志 IAM 策略”。
- 创建包含详细 CloudWatch 监控的 Amazon EC2 启动模板，然后在创建 AWS Batch 计算环境时指定该模板。您也可以在现有映像上安装 CloudWatch代理，然后在 AWS Batch 首次运行向导中指定映像。
- （可选）配置 awslogs 驱动程序。可以添加用于更改 EC2 和 Fargate 资源默认行为的参数。有关更多信息，请参阅 the section called “使用 awslogs 日志驱动程序”。

添加 CloudWatch 日志 IAM 策略

在您的任务可以向日志发送日志数据和详细指标之前，您必须创建使用 CloudWatch 日 CloudWatch 志 API 的 IAM 策略。创建 IAM policy 后，将其附加到`ecsInstanceRole`角色。

> **Note**
>
> 如果该`ECS-CloudWatchLogs`策略未附加到该`ecsInstanceRole`角色，则仍可以将基本指标发送到 CloudWatch Logs。但是，基本指标不包括日志数据或详细指标，例如可用磁盘空间。

AWS Batch 计算环境使用 Amazon EC2 资源。使用 AWS Batch 首次运行向导创建计算环境时，AWS Batch 会创建`ecsInstanceRole`角色并使用该角色配置环境。

如果您未使用首次运行向导，则可以在 AWS Command Line Interface 或 AWS Batch API 中创建计算环境时指定`ecsInstanceRole`角色。有关更多信息，请参阅AWS CLI 命令参考或 AWS Batch API 参考。

创建**ECS-CloudWatchLogs** IAM policy

1. 打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 选择创建策略。
4. 选择 JSON，然后输入以下策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

5. 选择下一步：标签。
6. （可选）对于添加标签，选择添加标签以向策略添加标签。
7. 选择下一步：查看。
8. 在查看策略页面上，对于名称，输入**ECS-CloudWatchLogs**；然后输入可选的描述。
9. 选择 创建策略。

将**ECS-CloudWatchLogs**策略附加到**ecsInstanceRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。

3. 选择`ecsInstanceRole`。如果角色不存在，请按照Amazon ECS 实例角色中的程序来创建角色。
4. 选择添加权限，然后选择附加策略。
5. 选择 ECS-CloudWatch Logs 策略，然后选择附加策略。

安装和配置代 CloudWatch 理

您可以创建包含 CloudWatch 监控功能的 Amazon EC2 启动模板。有关更多信息，请参阅 Amazon EC2 用户指南中的通过启动模板启动实例和高级详情。

您也可以在现有 Amazon EC2 AMI 上安装 CloudWatch 代理，然后在 AWS Batch 首次运行向导中指定映像。有关更多信息，请参阅安装 CloudWatch 代理和入门 AWS Batch。

> **Note**
>
> AWS Fargate 资源不支持启动模板。

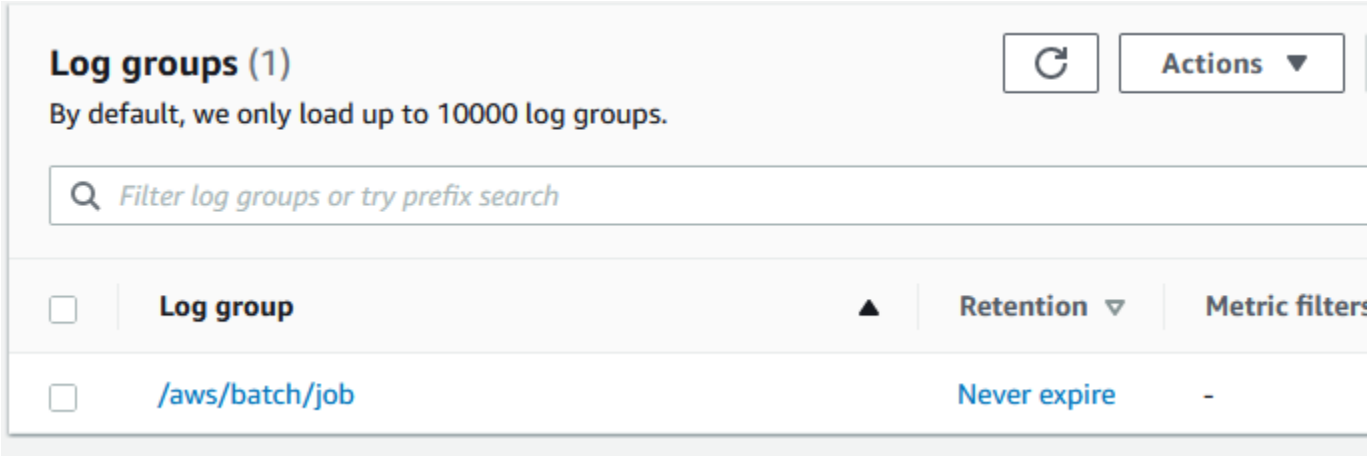
查看 CloudWatch 日志

您可以在中查看和搜索 CloudWatch 日志日志 AWS Management Console。

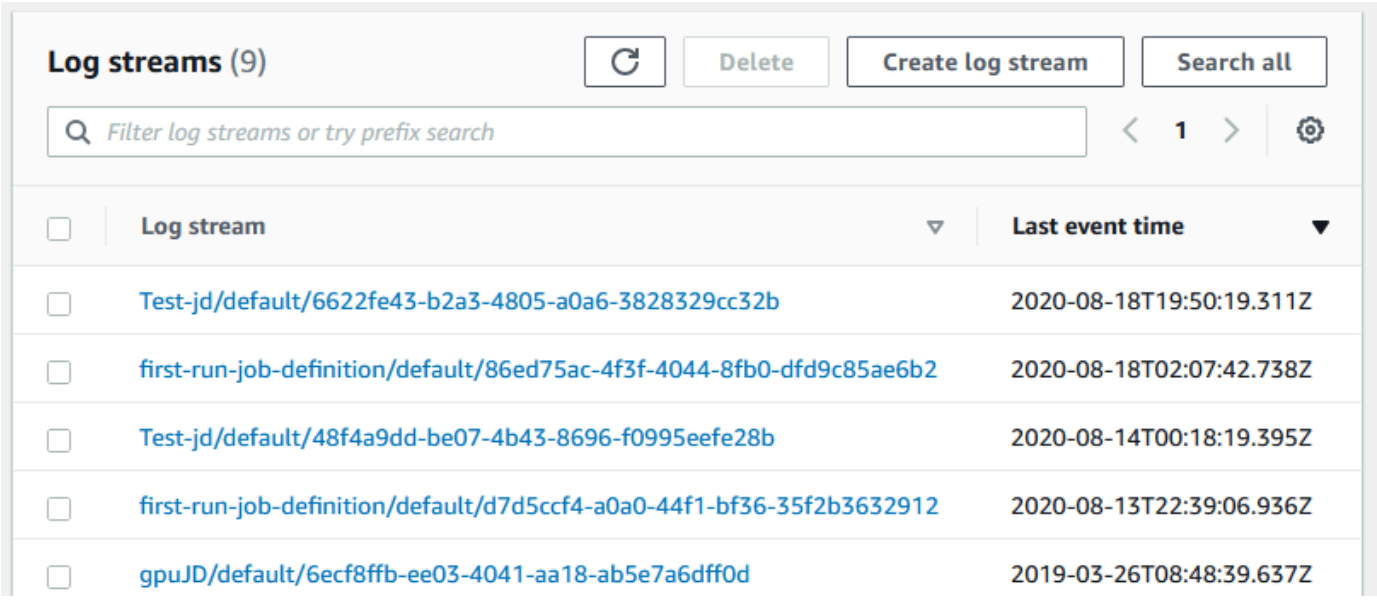
> **Note**
>
> 数据可能需要几分钟才能显示在 “ CloudWatch 日志” 中。

查看您的 CloudWatch 日志数据

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中选择日志，然后选择日志组。

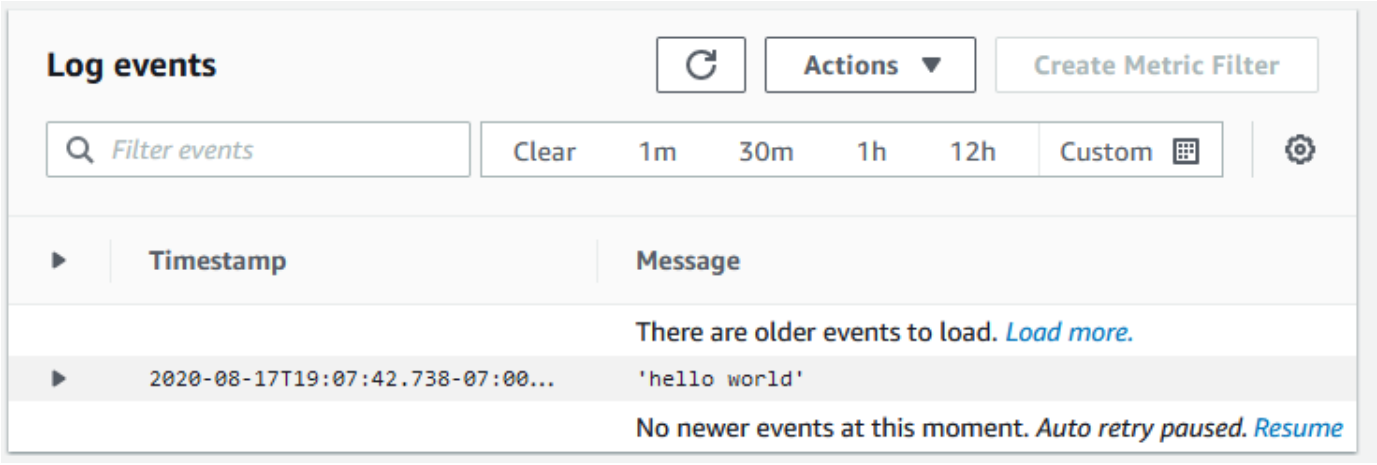


3. 选择要查看的日志组。



4. 选择要查看的日志流。默认情况下，数据流由作业名称的前 200 个字符和 Amazon ECS 任务 ID 进行标识。

> **Tip**
>
> 要下载日志流数据，请选择操作。


Log events
Actions
Create Metric Filter
Filter events
Clear
1m
30m
1h
12h
Custom
Timestamp
Message
There are older events to load. Load more.
2020-08-17T19:07:42.738-07:00...
'hello world'
No newer events at this moment. Auto retry paused. Resume

使用 CloudWatch Logs 监控 Amazon EKS 作业的 AWS Batch

您可以使用 Amazon CloudWatch Logs 来监控、存储和查看所有日志文件。使用 CloudWatch Logs，您可以搜索、筛选和分析来自多个来源的日志数据。

您可以下载 AWS 获取 Fluent Bit 映像，其中包含一个插件，用于监视 CloudWatch Logs 中 Amazon EKS 作业的 AWS Batch。Fluent Bit 是一个开源的日志处理器和转发器，其与 Docker 和 Kubernetes 兼容。我们建议您使用 Fluent Bit 作为日志路由器，因为它的资源密集度低于 Fluentd。有关更多信息，请参阅为 Fluent Bit 映像使用 AWS。

先决条件

将 `CloudWatchAgentServerPolicy` 策略附加到 Worker 节点的 AWS Identity and Access Management 策略。有关更多信息，请参阅先决条件。

安装 AWS 获取 Fluent Bit

有关如何安装 AWS 获取 Fluent Bit 和创建 CloudWatch 群组的说明，请参阅设置 Fluent Bit 或 CloudWatch 代理和 Fluent Bit 的快速入门。

> **Tip**
>
> 请记住，Fluent Bit 在 AWS Batch 节点上占用 5 CPU 和 100 MB 的内存。这会减少 AWS Batch 作业的总可用容量。在确定工作规模时，请考虑这一点。

为 AWS Batch 节点开启 Fluent Bit

要确保 Fluent Bit 日志 DaemonSet 在 AWS Batch 托管节点上运行，请修改 Fluent Bit DaemonSet 的容忍度：

```
tolerations:
- key: "batch.amazonaws.com/batch-node"
  operator: "Exists"
```

AWS Batch CloudWatch Container Insights

CloudWatch Container Insights 收集、汇总和总结来自AWS Batch计算环境和作业的指标和日志。指标包括 CPU、内存、磁盘和网络利用率。可以将这些指标添加到 CloudWatch 控制面板中。

运行数据是作为性能日志事件收集的。它们是使用结构化 JSON 模式的条目，该架构允许批量提取和存储高基数数据。CloudWatch 利用这些数据在计算环境和作业级别创建更高级别的汇总指标，作为 CloudWatch 指标。有关更多信息，请参阅《Amazon CloudWatch 用户指南》中的 Amazon ECS 的 Container Insights Structured Logs。

> **Important**
>
> CloudWatch Container Insights 由 CloudWatch 按照自定指标收费。有关更多信息，请参阅 Amazon CloudWatch Events 定价。

开启 Container Insights

您可以为AWS Batch计算环境启用 Container Insights。

1. 打开AWS Batch控制台。
2. 选择计算环境。
3. 选择所需的计算环境。
4. 对于 Container Insights，打开计算环境的 Container Insights。

> **Tip**
>
> 可以选择一个默认时间间隔来汇总指标，也可以创建一个自定义时间间隔。

默认情况下，会显示以下指标。有关 Amazon ECS Container Insights 指标的完整列表，请参阅《Amazon CloudWatch 用户指南》中的 Amazon ECS Container Insights 指标。

- **`JobCount`**— 在计算环境中运行的作业数量。
- **`ContainerInstanceCount`**— 运行 Amazon ECS 代理并在计算环境中注册的 Amazon Elastic Compute Cloud 实例的数量。

- **MemoryReserved**— 计算环境作业预留的内存。只有在作业定义中定义了内存预留的作业才会收集此指标。
- **MemoryUtilized**— 计算环境作业正在使用的内存。只有在作业定义中定义了内存预留的作业才会收集此指标。
- **CpuReserved**— 计算环境任务预留的 CPU 单元。只有在作业定义中定义了 CPU 预留的作业才会收集此指标。
- **CpuUtilized**— 计算环境中作业使用的 CPU 单元。只有在作业定义中定义了 CPU 预留的作业才会收集此指标。
- **NetworkRxBytes**— 收到的字节数。此指标仅适用于使用awsvpc或桥接网络模式的作业中的容器。
- **NetworkTxBytes**— 传输的字节数。此指标仅适用于使用awsvpc或桥接网络模式的作业中的容器。
- **StorageReadBytes**— 从存储中读取的字节数。
- **StorageWriteBytes**— 写入存储空间的字节数。

使用 AWS CloudTrail 记录 AWS Batch API 调用

AWS Batch 与 AWS CloudTrail 集成，后者是在 AWS 中记录用户、角色或 AWS Batch 服务所执行操作的服务。CloudTrail 将 AWS Batch 的所有 API 调用作为事件捕获。捕获的调用包含来自 AWS Batch 控制台和代码的 AWS Batch API 操作调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 AWS Batch 的事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 AWS Batch 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

CloudTrail 中的 AWS Batch 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 AWS Batch 中发生活动时，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 AWS Batch 的事件），请创建跟踪。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时，此跟踪应用于所有 AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）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。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 AWS Batch 操作都由 CloudTrail 记录，并记录在https://docs.aws.amazon.com/batch/latest/APIReference/中。例如，对 `SubmitJob`、`ListJobs` 和 `DescribeJobs` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。

- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 AWS Batch 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 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、操作的日期和时间、请求参数等方面的信息。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 `CreateComputeEnvironment` 操作。


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2017-12-20T00:48:46Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2017-12-20T00:48:46Z",
  "eventSource": "batch.amazonaws.com",
  "eventName": "CreateComputeEnvironment",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
```

```
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "computeResources": {
      "subnets": [
        "subnet-5eda8e04"
      ],
      "tags": {
        "testBatchTags": "CLI testing CE"
      },
      "desiredvCpus": 0,
      "minvCpus": 0,
      "instanceTypes": [
        "optimal"
      ],
      "securityGroupIds": [
        "sg-aba9e8db"
      ],
      "instanceRole": "ecsInstanceRole",
      "maxvCpus": 128,
      "type": "EC2"
    },
    "state": "ENABLED",
    "type": "MANAGED",
    "computeEnvironmentName": "Test"
  },
  "responseElements": {
    "computeEnvironmentName": "Test",
    "computeEnvironmentArn": "arn:aws:batch:us-east-1:012345678910:compute-environment/
Test"
  },
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```

创建虚拟私有云

计算环境中的计算资源需要外部网络访问权限以便与 AWS Batch 和 Amazon ECS 服务端点进行通信。但是，您可能有想要在私有子网中运行的作业。创建带有公有和私有子网的 VPC 可为您提供在公有子网或私有子网中运行作业的灵活性。

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。本主题提供了 Amazon VPC 向导的链接以及可供选择的选项列表。

创建 VPC

有关如何创建 Amazon VPC 的信息，请参阅 《Amazon VPC 用户指南》 中的 仅创建 VPC 并使用下表以确定要选择的选项。

选项	值
要创建的资源	仅限 VPC
名称	可以选择为您的 VPC 提供名称。
IPv4 CIDR 块	IPv4 CIDR 手动输入 CIDR 块大小必须在 /16 和 /28 之间。
IPv6 CIDR 块	无 IPv6 CIDR 块
租期	默认值

有关 Amazon VPC 的更多信息，请参阅 Amazon VPC 用户指南中的什么是Amazon VPC？。

后续步骤

在创建 VPC 后，您应考虑以下后续步骤：

- 如果您的公有和私有资源需要入站网络访问权限，则为这些资源创建安全组。有关更多信息，请参阅《Amazon VPC 用户指南》中的使用安全组。
- 创建一个可将计算资源启动到您的新 VPC 中的 AWS Batch 托管计算环境。有关更多信息，请参阅创建计算环境。如果您在 AWS Batch 控制台中使用了计算环境创建向导，则可以指定您刚刚创建的 VPC 以及要将您的实例启动到的公有或私有子网。
- 创建要映射到您的新计算环境的 AWS Batch 作业队列。有关更多信息，请参阅创建作业队列。
- 创建要用于运行您的任务的任务定义。有关更多信息，请参阅创建单节点作业定义 。
- 将带有任务定义的任务提交到您的新的任务队列。此任务将落到您使用新 VPC 和子网创建的计算环境中。有关更多信息，请参阅提交作业。

安全性 AWS Batch

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模型将其描述为云的安全性和云中的安全性。

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 AWS Batch，请参阅按合规计划划分的范围内的AWSAWS 服务按合规计划。
- 云中的安全性：您的责任由您使用的 AWS 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 AWS Batch。以下主题向您介绍如何进行配置 AWS Batch 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 AWS Batch 资源。

主题

- 适用于 Identity and Access Managem AWS Batch
- AWS Batch 使用接口端点进行访问
- 合规性验证 AWS Batch
- 中的基础设施安全 AWS Batch

适用于 Identity and Access Managem AWS Batch

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（拥有权限）使用 AWS Batch 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证

- 使用策略管理访问
- 如何 AWS Batch 与 IAM 配合使用
- AWS Batch 执行 IAM 角色
- 基于身份的策略示例 AWS Batch
- 防止跨服务混淆座席
- 对 AWS Batch 身份和访问进行故障排除
- 将服务相关角色用于 AWS Batch
- AWS 的托管策略 AWS Batch

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 AWS Batch。

服务用户-如果您使用 AWS Batch 服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多 AWS Batch 功能来完成工作时，你可能需要额外的权限。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 AWS Batch中的特征，请参阅 对 AWS Batch 身份和访问进行故障排除。

服务管理员-如果您负责公司的 AWS Batch 资源，则可能拥有完全访问权限 AWS Batch。您的工作是确定您的服务用户应访问哪些 AWS Batch 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。要详细了解您的公司如何将 IAM 与配合使用 AWS Batch，请参阅如何 AWS Batch 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 AWS Batch的访问权限的详细信息。要查看您可以在 IAM 中使用的 AWS Batch 基于身份的策略示例，请参阅。基于身份的策略示例 AWS Batch

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。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。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。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 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。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，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

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 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，对于某些资源 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 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 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 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当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-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon 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证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该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

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。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。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。管理员随后可以向角色添加 IAM 策略，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。拥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。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括

AWS 托管策略和客户托管策略。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。委托人可以包括账户、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 IAM 中的 AWS 托管策略。

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。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF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。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体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，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）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。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

如何 AWS Batch 与 IAM 配合使用

在使用 IAM 管理访问权限之前 AWS Batch，请先了解有哪些 IAM 功能可供使用 AWS Batch。

您可以搭配使用的 IAM 功能 AWS Batch

IAM 功能	AWS Batch 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACL	否
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	是
服务相关角色	是

要全面了解 AWS Batch 以及其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

基于身份的策略 AWS Batch

支持基于身份的策略	是

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。

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。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 AWS Batch的基于身份的策略示例

要查看 AWS Batch 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS Batch

的政策行动 AWS Batch

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。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。

要查看 AWS Batch 操作列表，请参阅《服务授权参考》 AWS Batch中定义的操作。

正在执行的策略操作在操作前 AWS Batch 使用以下前缀：

```
batch
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "batch:action1",
      "batch: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": "batch:Describe*"
```

要查看 AWS Batch 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS Batch

的政策资源 AWS Batch

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。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": "*"
```

要查看 AWS Batch 资源类型及其 ARN 的列表，请参阅《服务授权参考》 AWS Batch中的 “由定义的资源”。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 AWS Batch定义的操作。

要查看 AWS Batch 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS Batch

AWS Batch的策略条件键

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。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。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 AWS Batch 条件键列表，请参阅《服务授权参考》 AWS Batch中的条件密钥。要了解您可以使用条件键的操作和资源，请参阅操作定义者 AWS Batch。

要查看 AWS Batch 基于身份的策略的示例，请参阅。基于身份的策略示例 AWS Batch

基于属性的访问控制 (ABAC) AWS Batch

支持 ABAC（策略中的标签）	是

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）和许多 AWS 资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

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。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

使用临时凭证和 AWS Batch

支持临时凭证	是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

AWS Batch的跨服务主体权限

支持转发访问会话 (FAS)	是

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当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。

的服务角色 AWS Batch

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 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。

> **Warning**
>
> 更改服务角色的权限可能会中断 AWS Batch 功能。只有在 AWS Batch 提供指导时，才能编辑服务角色。

的服务相关角色 AWS Batch

支持服务相关角色	是

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

AWS Batch 执行 IAM 角色

执行角色授予 Amazon ECS 容器和 AWS Fargate 代理代表您 AWS 进行 API 调用的权限。

> **Note**
>
> 该执行角色由 Amazon ECS 容器代理版本 1.16.0 和更高版本支持。

执行 IAM 角色是必需的，具体取决于任务的要求。您可以将多个执行角色用于与您的账户关联的服务不同目的。

> **Note**
>
> 有关 Amazon ECS 实例角色的信息，请参阅 Amazon ECS 实例角色。有关服务角色的更多信息，请参阅如何 AWS Batch 与 IAM 配合使用。

Amazon ECS 提供 AmazonECSTaskExecutionRolePolicy 托管策略。该策略包含上述常见使用案例所需的权限。对于特殊使用案例，可能需要向您的执行角色添加内联策略，这些策略概述如下。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

您可以使用以下过程检查并确定您的账户是否已拥有执行角色并且已附加托管 IAM policy（如果需要）。

在 IAM 控制台中检查 **ecsTaskExecutionRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 在角色列表中搜索 ecsTaskExecutionRole。如果找不到角色，请参阅创建执行 IAM 角色。如果找到该角色，请选择角色以查看附加的策略。
4. 在“权限”选项卡上，验证 AmazonECS TaskExecution RolePolicy 托管策略是否已附加到该角色。如果附加该策略，则将正确配置 执行角色。否则，请执行以下子步骤来附加策略。
   a. 选择添加权限，然后选择附加策略。
   b. 搜索亚马逊TaskExecutionRolePolicy。
   c. 选中 Amazonec s 政策左侧的复选框并选择附加TaskExecutionRolePolicy政策。
5. 选择 Trust Relationships（信任关系）。
6. 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，则该角色的配置正确。如果信任关系不匹配，请选择编辑信任策略，输入以下策略，然后选择更新策略。

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

创建执行 IAM 角色

如果您的账户尚未具有执行角色，请使用以下步骤创建角色。

创建 **ecsTaskExecutionRole** IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles（角色）。
3. 选择 创建角色。
4. 对于 可信实体类型，选择 AWS 服务。
5. 对于服务或使用案例，选择 EC2。然后再次选择 EC2。
6. 选择下一步。
7. 要查看权限政策，请搜索亚马逊 ECS TaskExecution RolePolicy。
8. 选中 AmazonECS TaskExecution RolePolicy 政策左侧的复选框，然后选择 “下一步”。
9. 对于角色名称，输入 `ecsTaskExecutionRole`，然后选择创建角色。

基于身份的策略示例 AWS Batch

默认情况下，用户和角色没有创建或修改 AWS Batch 资源的权限。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。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。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅 IAM 用户指南中的 创建 IAM 策略。

有关由定义的操作和资源类型的详细信息 AWS Batch，包括每种资源类型的 ARN 格式，请参阅《服务授权参考》 AWS Batch中的操作、资源和条件密钥。

主题

- 策略最佳实践
- 使用控制 AWS Batch 台
- 允许用户查看他们自己的权限

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 AWS Batch 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：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管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。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。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。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。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。

使用控制 AWS Batch 台

要访问 AWS Batch 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您的 AWS Batch 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 AWS Batch 控制台，还需要将 AWS Batch `ConsoleAccess`或`ReadOnly` AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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": "*"
        }
    ]
}
```

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况， AWS 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

我们建议在资源策略中使用aws:SourceArn和aws:SourceAccount全局条件上下文密钥来限制为资源 AWS Batch 提供其他服务的权限。如果 aws:SourceArn 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文密钥来限制权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 aws:SourceArn 值，则 aws:SourceAccount 值和 aws:SourceArn 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。如果您只希望将一个资源与跨服务访问相关联，请使用 aws:SourceArn。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 aws:SourceAccount。

的值aws:SourceArn必须是 AWS Batch 存储的资源。

防范混淆代理问题最有效的方法是使用 aws:SourceArn 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的 aws:SourceArn 全局上下文条件键。例如，arn:aws:*servicename*:*:*123456789012*:*。

以下示例说明了如何使用中的aws:SourceArn和aws:SourceAccount全局条件上下文键 AWS Batch 来防止出现混淆的副手问题。

示例 1：仅用于访问一个计算环境的角色

以下角色只能用于访问一个计算环境。必须将作业名称指定为 *，因为作业队列可以与多个计算环境相关联。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "batch.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": [
            "arn:aws:batch:us-east-1:123456789012:compute-environment/testCE",
            "arn:aws:batch:us-east-1:123456789012:job/*"
          ]
        }
      }
    }
  ]
}
```

示例 2：访问多个计算环境的角色

以下角色可用于访问多个计算环境。必须将作业名称指定为 *，因为作业队列可以与多个计算环境相关联。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "batch.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": [
```

```
            "arn:aws:batch:us-east-1:123456789012:compute-environment/*",
            "arn:aws:batch:us-east-1:123456789012:job/*"
          ]
        }
      }
    }
  ]
}
```

对 AWS Batch 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 AWS Batch 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 AWS Batch中执行操作
- 我无权执行 iam：PassRole
- 我想允许 AWS 账户之外的人访问我的 AWS Batch 资源

我无权在 AWS Batch中执行操作

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 `mateojackson` 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `batch:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 batch:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `batch:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。有关授予角色传递权限的更多信息，请参阅向用户授予将角色传递给 AWS 服务的权限。

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 AWS Batch。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的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 AWS Batch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。

我想允许 AWS 账户之外的人访问我的 AWS Batch 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：

- 要了解是否 AWS Batch 支持这些功能，请参阅如何 AWS Batch 与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）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。

将服务相关角色用于 AWS Batch

AWS Batch 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种与之直接关联的 IAM 角色的独特类型。 AWS Batch服务相关角色由服务预定义 AWS Batch ，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色使设置变得 AWS Batch 更加容易，因为您不必手动添加必要的权限。 AWS Batch 定义其服务相关角色的权限，除非另有定义，否则 AWS Batch 只能担任其角色。定义的权限包括信任策略和权限策略，而且权限策略不能附加到任何其它 IAM 实体。

> **Note**
>
> 执行以下任一操作为 AWS Batch 计算环境指定服务角色。
>
> - 对服务角色使用空字符串。这样就可以 AWS Batch 创建服务角色了。
> - 采用以下格式指定服务角色：`arn:aws:iam::account_number:role/aws-service-role/batch.amazonaws.com/AWSServiceRoleForBatch`。
>
> 有关更多信息，请参阅the section called “角色名称或 ARN 不正确”《 AWS Batch 用户指南》。

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 AWS Batch 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的AWS 服务并查找服务相关角色列中显示为是的服务。选择是和链接，查看该服务的服务相关角色文档。

的服务相关角色权限 AWS Batch

AWS Batch 使用名AWSServiceRoleForBatch为的服务相关角色。该AWSServiceRoleForBatch角色 AWS Batch 允许代表您创建和管理 AWS 资源。

AWSServiceRoleForBatch服务相关角色信任`batch.amazonaws.com`服务主体代入该角色。

名为的 IAM 策略BatchServiceRolePolicy AWS Batch 允许对特定资源完成以下操作：

- `autoscaling`— AWS Batch 允许创建和管理 Amazon EC2 Auto Scaling 资源。 AWS Batch 为大多数计算环境创建和管理 Amazon EC2 Auto Scaling 组。
- `ec2`— AWS Batch 允许控制 Amazon EC2 实例的生命周期以及创建和管理启动模板和标签。 AWS Batch 为某些 EC2 竞价计算环境创建和管理 EC2 竞价型队列请求。
- `ecs`- AWS Batch 允许创建和管理 Amazon ECS 集群、任务定义和任务执行任务。
- `eks`- AWS Batch 允许描述用于验证的 Amazon EKS 集群资源。
- `iam`-允许 AWS Batch 验证所有者提供的角色并将其传递给 Amazon EC2、Amazon EC2 Auto Scaling 和 Amazon ECS。
- `logs`— AWS Batch 允许创建和管理 AWS Batch 作业的日志组和日志流。

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。

为创建服务相关角色 AWS Batch

您无需手动创建服务相关角色。当你 CreateComputeEnvironment 在 AWS Management Console AWS CLI、或 AWS API 中并且没有为`serviceRole`参数指定值时， AWS Batch 会为你创建服务相关角色。

> **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。另外，如果您在 2021 年 3 月 10 日之前使用该 AWS Batch 服务，即该服务开始支持服务相关角色，则在您的账户中 AWS Batch 创建了该 AWSServiceRoleForBatch 角色。要了解更多信息，请参阅我的 IAM 账户中的新角色。

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。当你时 CreateComputeEnvironment， AWS Batch 会再次为你创建服务相关角色。

编辑的服务相关角色 AWS Batch

使用 AWS Batch，您无法编辑 AWSServiceRoleForBatch 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。

允许 IAM 实体编辑 AWSServiceRoleForBatch 服务相关角色的描述

向该权限策略添加以下声明。这允许 IAM 实体编辑服务相关角色的描述。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:UpdateRoleDescription"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/batch.amazonaws.com/
AWSServiceRoleForBatch",
    "Condition": {"StringLike": {"iam:AWSServiceName": "batch.amazonaws.com"}}
}
```

删除的服务相关角色 AWS Batch

如果不再需要使用某个需要服务相关角色的特征或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

允许 IAM 实体删除 AWSServiceRoleForBatch 服务相关角色

向该权限策略添加以下声明。这允许 IAM 实体删除服务相关角色。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/batch.amazonaws.com/
AWSServiceRoleForBatch",
    "Condition": {"StringLike": {"iam:AWSServiceName": "batch.amazonaws.com"}}
}
```

清除服务相关角色

在使用 IAM 删除服务相关角色之前，必须先确认该角色没有活动会话，然后删除单个分区中所有 AWS 区域中使用该角色的所有 AWS Batch 计算环境。

检查服务相关角色是否有活动会话

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色，然后选择 AWSServiceRoleForBatch 名称（不是复选框）。
3. 在 Summary 页面上，选择 Access Advisor，查看服务相关角色的近期活动。

> **Note**
>
> 如果您不知道 AWS Batch 是否在使用该 AWSServiceRoleForBatch 角色，可以尝试删除该角色。如果服务正在使用该角色，则该角色将无法删除。您可以查看正在使用该角色的区域。如果该角色已被使用，则您必须等待会话结束，然后才能删除该角色。无法撤销服务相关角色对会话的权限。

移除 AWSServiceRoleForBatch 服务相关角色使用的 AWS Batch 资源

必须先删除所有 AWS 区域中使用该 AWSServiceRoleForBatch 角色的所有 AWS Batch 计算环境，然后才能删除该 AWSServiceRoleForBatch 角色。

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择计算环境。
4. 选择计算环境。
5. 选择 Disable（禁用）。等待状态变为已禁用。
6. 选择计算环境。
7. 选择删除。通过选择删除计算环境来确认您要删除计算环境。
8. 对所有区域中使用服务相关角色的所有计算环境重复步骤 1—7。

在 IAM 中删除服务相关角色 (控制台)

您可以使用 IAM 控制台删除服务相关角色。

删除服务相关角色 (控制台)

1. 登录 AWS Management Console 并打开 IAM 控制台，网址为 https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。然后选中旁边的复选框 AWSServiceRoleForBatch，而不是名称或行本身。
3. 选择删除角色。
4. 在确认对话框中，查看上次访问服务数据，该数据显示每个选定角色上次访问 AWS 服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续，请选择 Yes, Delete 以提交服务相关角色进行删除。
5. 监视 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色删除是异步的，因此，在您提交角色进行删除后，删除任务可能成功，也可能失败。
   - 如果任务成功，则角色将从列表中删除，并会在页面顶部显示成功通知。
   - 如果任务失败，您可以从通知中选择 View details 或 View Resources 以了解删除失败的原因。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以清除资源并再次提交删除。

> **Note**
>
> 您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。

- 如果任务失败，并且通知不包含资源列表，则服务可能不会返回该信息。要了解如何清除该服务的资源，请参阅 使用 IAM 的AWS 服务。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

在 IAM 中删除服务相关角色 (AWS CLI)

您可以使用中的 IAM 命令 AWS Command Line Interface 删除服务相关角色。

删除服务相关角色（CLI）

1. 如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `deletion-task-id` 以检查删除任务的状态。键入以下命令以提交服务相关角色的删除请求：

```
$ aws iam delete-service-linked-role --role-name AWSServiceRoleForBatch
```

2. 使用以下命令以检查删除任务的状态：

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

删除任务的状态可能是 NOT_STARTED、IN_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以清除资源并再次提交删除。

> **Note**
>
> 您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源，其中一些资源。或者，它可能不会报告任何资源。要了解如何为不报告

任何资源的服务清除资源，请参阅 AWS 使用 IAM 的服务。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

在 IAM 中删除服务相关角色 (AWS API)

您可以使用 IAM API 删除服务相关角色。

删除服务相关角色（API）

1. 要提交与服务相关的名册的删除请求，请致电DeleteServiceLinkedRole。在请求中，指定 AWSServiceRoleForBatch 角色名称。

   如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `DeletionTaskId` 以检查删除任务的状态。

2. 要查看删除状态，请致电GetServiceLinkedRoleDeletionStatus。在请求中，指定 `DeletionTaskId`。

   删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以清除资源并再次提交删除。

   > **Note**
   >
   > 您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。要了解如何为不报告任何资源的服务清除资源，请参阅 使用 IAM 的AWS 服务。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

AWS Batch 服务相关角色支持的区域

AWS Batch 支持在提供服务的所有区域中使用服务相关角色。有关更多信息，请参阅 AWS Batch 端点。

AWS 的托管策略 AWS Batch

您可以使用 AWS 托管策略来简化团队和已配置 AWS 资源的身份访问管理。 AWS 托管政策涵盖各种常见用例，默认情况下可在您的 AWS 账户中使用，并以您的名义进行维护和更新。您无法更改 AWS 托管策略中的权限。如果您需要更大的灵活性，也可以选择创建 IAM 客户管理型策略。这样，您就可以为团队预配置的资源提供他们所需的确切权限。

有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的AWS 托管策略。

AWS 服务代表您维护和更新 AWS 托管政策。 AWS 服务会定期向 AWS 托管策略添加其他权限。AWS 当有新功能启动或操作可用时，托管策略很可能会更新。此类更新会自动影响附加策略的所有身份（用户、组和角色）。但是，它们不会移除权限或破坏您的现有权限。

此外，还 AWS 支持跨多个服务的工作职能的托管策略。例如，`ReadOnlyAccess` AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动一项新功能时， AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的适用于工作职能的AWS 托管式策略。

AWS 托管策略：BatchServiceRolePolicy

BatchServiceRolePolicy托管 IAM 策略由`AWSServiceRoleForBatch`服务相关角色使用。这 AWS Batch 允许您代表您执行操作。您不能将此策略附加到您的 IAM 实体。有关更多信息，请参阅 将服务相关角色用于 AWS Batch。

此策略 AWS Batch 允许对特定资源完成以下操作：

- `autoscaling`— AWS Batch 允许创建和管理 Amazon EC2 Auto Scaling 资源。 AWS Batch 为大多数计算环境创建和管理 Amazon EC2 Auto Scaling 组。
- `ec2`— AWS Batch 允许控制 Amazon EC2 实例的生命周期以及创建和管理启动模板和标签。 AWS Batch 为某些 EC2 竞价计算环境创建和管理 EC2 竞价型队列请求。
- `ecs`- AWS Batch 允许创建和管理 Amazon ECS 集群、任务定义和任务执行任务。
- `eks`- AWS Batch 允许描述用于验证的 Amazon EKS 集群资源。

- `iam`-允许 AWS Batch 验证所有者提供的角色并将其传递给 Amazon EC2、Amazon EC2 Auto Scaling 和 Amazon ECS。
- `logs`— AWS Batch 允许创建和管理 AWS Batch 作业的日志组和日志流。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSBatchPolicyStatement1",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeImages",
                "ec2:DescribeImageAttribute",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSpotFleetInstances",
                "ec2:DescribeSpotFleetRequests",
                "ec2:DescribeSpotPriceHistory",
                "ec2:DescribeSpotFleetRequestHistory",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:RequestSpotFleet",
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "eks:DescribeCluster",
                "ecs:DescribeClusters",
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTaskDefinition",
                "ecs:DescribeTasks",
                "ecs:ListClusters",
                "ecs:ListContainerInstances",
                "ecs:ListTaskDefinitionFamilies",
                "ecs:ListTaskDefinitions",
```

```
                "ecs:ListTasks",
                "ecs:DeregisterTaskDefinition",
                "ecs:TagResource",
                "ecs:ListAccountSettings",
                "logs:DescribeLogGroups",
                "iam:GetInstanceProfile",
                "iam:GetRole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AWSBatchPolicyStatement2",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/batch/job*"
        },
        {
            "Sid": "AWSBatchPolicyStatement3",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/batch/job*:log-stream:*"
        },
        {
            "Sid": "AWSBatchPolicyStatement4",
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateOrUpdateTags"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/AWSBatchServiceTag": "false"
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement5",
            "Effect": "Allow",
            "Action": "iam:PassRole",
```

```
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement6",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com",
                        "autoscaling.amazonaws.com",
                        "ecs.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement7",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/AWSBatchServiceTag": "false"
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement8",
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances",
                "ec2:CancelSpotFleetRequests",
                "ec2:ModifySpotFleetRequest",
                "ec2:DeleteLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:ResourceTag/AWSBatchServiceTag": "false"
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement9",
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:DeleteLaunchConfiguration"
            ],
            "Resource":
"arn:aws:autoscaling:*:*:launchConfiguration:*:launchConfigurationName/AWSBatch*"
        },
        {
            "Sid": "AWSBatchPolicyStatement10",
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:SuspendProcesses",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:TerminateInstanceInAutoScalingGroup"
            ],
            "Resource":
"arn:aws:autoscaling:*:*:autoScalingGroup:*:autoScalingGroupName/AWSBatch*"
        },
        {
            "Sid": "AWSBatchPolicyStatement11",
            "Effect": "Allow",
            "Action": [
                "ecs:DeleteCluster",
```

```
                "ecs:DeregisterContainerInstance",
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:StopTask"
            ],
            "Resource": "arn:aws:ecs:*:*:cluster/AWSBatch*"
        },
        {
            "Sid": "AWSBatchPolicyStatement12",
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:StopTask"
            ],
            "Resource": "arn:aws:ecs:*:*:task-definition/*"
        },
        {
            "Sid": "AWSBatchPolicyStatement13",
            "Effect": "Allow",
            "Action": [
                "ecs:StopTask"
            ],
            "Resource": "arn:aws:ecs:*:*:task/*/*"
        },
        {
            "Sid": "AWSBatchPolicyStatement14",
            "Effect": "Allow",
            "Action": [
                "ecs:CreateCluster",
                "ecs:RegisterTaskDefinition"
            ],
            "Resource": "*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/AWSBatchServiceTag": "false"
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement15",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
```

```
                "arn:aws:ec2:*::image/*",
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:launch-template/*",
                "arn:aws:ec2:*:*:placement-group/*",
                "arn:aws:ec2:*:*:capacity-reservation/*",
                "arn:aws:ec2:*:*:elastic-gpu/*",
                "arn:aws:elastic-inference:*:*:elastic-inference-accelerator/*",
                "arn:aws:resource-groups:*:*:group/*"
            ]
        },
        {
            "Sid": "AWSBatchPolicyStatement16",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "Null": {
                    "aws:RequestTag/AWSBatchServiceTag": "false"
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement17",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "RunInstances",
                        "CreateLaunchTemplate",
                        "RequestSpotFleet"
                    ]
                }
            }
```

```
        }
    ]
}
```

AWS 托管策略:AWSBatchServiceRole策略

名为的角色权限策略AWSBatchServiceRole AWS Batch 允许对特定资源完成以下操作：

AWSBatchServiceRole托管 IAM 策略通常由名为的角色使用 AWSBatchServiceRole，该策略包含以下权限。按照授予最低权限的标准安全建议，可以将AWSBatchServiceRole托管策略用作指导。如果您的用例不需要托管策略中授予的任何权限，请创建自定义策略并仅添加所需的权限。此 AWS Batch 托管策略和角色可用于大多数计算环境类型，但为了获得更好范围和更好的托管体验 less-error-prone，最好使用服务相关角色。

- `autoscaling`— AWS Batch 允许创建和管理 Amazon EC2 Auto Scaling 资源。 AWS Batch 为大多数计算环境创建和管理 Amazon EC2 Auto Scaling 组。
- `ec2`— AWS Batch 允许管理 Amazon EC2 实例的生命周期以及创建和管理启动模板和标签。 AWS Batch 为某些 EC2 竞价计算环境创建和管理 EC2 竞价型队列请求。
- `ecs`- AWS Batch 允许创建和管理 Amazon ECS 集群、任务定义和任务执行任务。
- `iam`-允许 AWS Batch 验证所有者提供的角色并将其传递给 Amazon EC2、Amazon EC2 Auto Scaling 和 Amazon ECS。
- `logs`— AWS Batch 允许创建和管理 AWS Batch 作业的日志组和日志流。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSBatchPolicyStatement1",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeImages",
                "ec2:DescribeImageAttribute",
```

```
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSpotFleetInstances",
                "ec2:DescribeSpotFleetRequests",
                "ec2:DescribeSpotPriceHistory",
                "ec2:DescribeSpotFleetRequestHistory",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:CreateLaunchTemplate",
                "ec2:DeleteLaunchTemplate",
                "ec2:RequestSpotFleet",
                "ec2:CancelSpotFleetRequests",
                "ec2:ModifySpotFleetRequest",
                "ec2:TerminateInstances",
                "ec2:RunInstances",
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:SuspendProcesses",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "ecs:DescribeClusters",
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTaskDefinition",
                "ecs:DescribeTasks",
                "ecs:ListAccountSettings",
                "ecs:ListClusters",
                "ecs:ListContainerInstances",
                "ecs:ListTaskDefinitionFamilies",
                "ecs:ListTaskDefinitions",
                "ecs:ListTasks",
                "ecs:CreateCluster",
                "ecs:DeleteCluster",
                "ecs:RegisterTaskDefinition",
                "ecs:DeregisterTaskDefinition",
                "ecs:RunTask",
```

```
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:UpdateContainerAgent",
                "ecs:DeregisterContainerInstance",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "iam:GetInstanceProfile",
                "iam:GetRole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AWSBatchPolicyStatement2",
            "Effect": "Allow",
            "Action": "ecs:TagResource",
            "Resource": [
                "arn:aws:ecs:*:*:task/*_Batch_*"
            ]
        },
        {
            "Sid": "AWSBatchPolicyStatement3",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement4",
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com",
                        "autoscaling.amazonaws.com",
                        "ecs.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AWSBatchPolicyStatement5",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "RunInstances"
                }
            }
        }
    ]
}
```

AWS 托管策略：AWSBatchFullAccess

该AWSBatchFullAccess策略授予 AWS Batch 操作对 AWS Batch 资源的完全访问权限。它还授予亚马逊 EC2、Amazon ECS、Amazon EKS 和 IAM 服务的描述和列出操作权限。 CloudWatch这样，IAM 身份（无论是用户还是角色）都可以查看代表他们创建的 AWS Batch 托管资源。最后，该策略还允许将选定的 IAM 角色传递给这些服务。

您可以附加AWSBatchFullAccess到您的 IAM 实体。 AWS Batch 还将此策略附加 AWS Batch 到允许代表您执行操作的服务角色。

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

```
      "Action":[
        "batch:*",
        "cloudwatch:GetMetricStatistics",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeVpcs",
        "ec2:DescribeImages",
        "ec2:DescribeLaunchTemplates",
        "ec2:DescribeLaunchTemplateVersions",
        "ecs:DescribeClusters",
        "ecs:Describe*",
        "ecs:List*",
        "eks:DescribeCluster",
        "eks:ListClusters",
        "logs:Describe*",
        "logs:Get*",
        "logs:TestMetricFilter",
        "logs:FilterLogEvents",
        "iam:ListInstanceProfiles",
        "iam:ListRoles"
      ],
      "Resource":"*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "iam:PassRole"
      ],
      "Resource":[
        "arn:aws:iam::*:role/AWSBatchServiceRole",
        "arn:aws:iam::*:role/service-role/AWSBatchServiceRole",
        "arn:aws:iam::*:role/ecsInstanceRole",
        "arn:aws:iam::*:instance-profile/ecsInstanceRole",
        "arn:aws:iam::*:role/iaws-ec2-spot-fleet-role",
        "arn:aws:iam::*:role/aws-ec2-spot-fleet-role",
        "arn:aws:iam::*:role/AWSBatchJobRole*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
```

```
      "Resource":"arn:aws:iam::*:role/*Batch*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "batch.amazonaws.com"
          }
      }
    }
  ]
}
```

AWS BatchAWS 托管策略的更新

查看 AWS Batch 自该服务开始跟踪这些更改以来 AWS 托管策略更新的详细信息。要获得有关此页面变更的自动提醒，请订阅 “ AWS Batch 文档历史记录” 页面上的 RSS feed。

更改	描述	日期
BatchServiceRolePolicy政策已更新	已更新，增加了对描述 Spot 队列请求历史记录和 Amazon EC2 Auto Scaling 活动的支持。	2023 年 12 月 5 日
AWSBatchServiceRole策略已添加	更新为添加语句 ID、向`ec2:DescribeSpotFleetRequestHistory` 和授予 AWS Batch 权限`autoscaling:DescribeScalingActivities` 。	2023 年 12 月 5 日
BatchServiceRolePolicy政策已更新	更新，增加了对描述 Amazon EKS 集群的支持。	2022 年 10 月 20 日
AWSBatchFullAccess政策已更新	更新，增加了对列出和描述 Amazon EKS 集群的支持。	2022 年 10 月 20 日

更改	描述	日期
BatchServiceRolePolicy政策已更新	更新，增加了对由 AWS Resource Groups 管理的 Amazon EC2 容量预留组的支持。有关更多信息，请参阅 Amazon EC2 用户指南中的使用容量预留组。	2022 年 5 月 18 日
BatchServiceRolePolicy并更新了AWSBatchServiceRole政策	更新后增加了对描述 Amazon EC2 中 AWS Batch 托管实例状态的支持，以便替换运行状况不佳的实例。	2021 年 12 月 6 日
BatchServiceRolePolicy政策已更新	更新，以增加对 Amazon EC2 中的置放群组、容量预留、弹性 GPU 和 Elastic Inference 资源的支持。	2021 年 3 月 26 日
BatchServiceRolePolicy策略已添加	借助AWSServiceRoleForBatch服务相关角色的BatchServiceRolePolicy托管策略，您可以使用由管理的服务相关角色。 AWS Batch有了此策略，您无需维护自己的角色即可在计算环境中使用。	2021 年 3 月 10 日
AWSBatchFullAccess-添加添加服务相关角色的权限	添加 IAM 权限以允许将AWSServiceRoleForBatch服务相关角色添加到账户。	2021 年 3 月 10 日
AWS Batch 开始跟踪更改	AWS Batch 开始跟踪其 AWS 托管策略的更改。	2021 年 3 月 10 日

AWS Batch 使用接口端点进行访问

您可以使用 AWS PrivateLink 在您的 VPC 和之间创建私有连接 AWS Batch。您可以像在 VPC 中一样访问 AWS Batch ，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 AWS Batch。

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。这些是请求者托管的网络接口，用作发往 AWS Batch的流量的入口点。

有关更多信息，请参阅 AWS PrivateLink 指南 中的 接口 VPC 端点。

的注意事项 AWS Batch

在为设置接口终端节点之前 AWS Batch，请查看AWS PrivateLink 指南中的接口端点属性和限制。

AWS Batch 支持通过接口端点调用其所有 API 操作。

在为设置接口 VPC 终端节点之前 AWS Batch，请注意以下注意事项：

- 使用 Fargate 资源启动类型的任务不需要 Amazon ECS 的接口 VPC 终端节点，但您可能需要以下几点中描述的 Amazon ECR、Secret CloudWatch s Manager 或 A AWS Batch mazon Logs 的接口 VPC 终端节点。
  - 要运行作业，您必须为 Amazon ECS 创建接口 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的接口 VPC 端点 (AWS PrivateLink)。
  - 要允许您的任务从 Amazon ECR 拉取私有映像，您必须为 Amazon ECR 创建接口 VPC 端点。有关更多信息，请参阅 Amazon Elastic Container Registry 用户指南中的接口 VPC 端点 (AWS PrivateLink)。
  - 要允许您的任务从 Secrets Manager 拉取敏感数据，您必须为 Secrets Manager 创建接口 VPC 端点。有关更多信息，请参阅 AWS Secrets Manager 用户指南中的将 Secrets Manager 与 VPC 端点结合使用。
  - 如果您的 VPC 没有 Internet 网关，并且您的任务使用`awslogs`日志驱动程序向日志发送日志信息，则必须为 CloudWatch 日志创建接口 VPC 终端节点。 CloudWatch 有关更多信息，请参阅 Amazon CloudWatch CloudWatch 日志用户指南中的将日志与接口 VPC 终端节点配合使用。
- 使用 EC2 资源的任务要求启动它们的容器实例运行 `1.25.1` 或更高版本的 Amazon ECS 容器代理。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》 中的 Amazon ECS 容器代理配置。
- VPC 端点当前不支持跨区域请求。确保在计划向 AWS Batch发出 API 调用的同一区域中创建端点。
- VPC 端点仅通过 Amazon Route 53 支持 Amazon 提供的 DNS。如果您希望使用自己的 DNS，可以使用条件 DNS 转发。有关更多信息，请参阅 Amazon VPC 用户指南中的 DHCP 选项集。
- 附加到 VPC 端点的安全组必须允许端口 443 上来自 VPC 的私有子网的传入连接。
- AWS Batch 不支持以下 VPC 接口终端节点 AWS 区域：
  - 亚太地区（大阪）(`ap-northeast-3`)
  - 亚太地区（雅加达）（`ap-southeast-3`）

为创建接口终端节点 AWS Batch

您可以创建用于 AWS Batch 使用 Amazon VPC 控制台或 AWS Command Line Interface (AWS CLI) 的接口终端节点。有关更多信息，请参阅《AWS PrivateLink 指南》中的创建接口端点。

AWS Batch 使用以下服务名称创建接口终端节点：

```
com.amazonaws.region.batch
```

例如：

```
com.amazonaws.us-east-2.batch
```

在aws-cn分区中，格式不同：

```
cn.com.amazonaws.region.batch
```

例如：

```
cn.com.amazonaws.cn-northwest-1.batch
```

如果您为接口终端节点启用私有 DNS，则 AWS Batch 可以使用其默认区域 DNS 名称向发出 API 请求。例如，batch.us-east-1.amazonaws.com。

有关更多信息，请参阅 AWS PrivateLink 指南中的通过接口端点访问服务。

为接口端点创建端点策略

端点策略是一种 IAM 资源，您可以将其附加到接口端点。默认终端节点策略允许 AWS Batch 通过接口终端节点进行完全访问。要控制允许从 VPC 访问 AWS Batch 的权限，请将自定义端点策略附加到接口端点。

端点策略指定以下信息：

- 可执行操作的主体（AWS 账户、用户和 IAM 角色）。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《AWS PrivateLink 指南》中的使用端点策略控制对服务的访问权限。

示例：用于 AWS Batch 操作的 VPC 终端节点策略

以下是自定义端点策略的示例。当您将此策略附加到接口终端节点时，它会授予所有委托人对所有资源 AWS Batch 执行所列操作的访问权限。

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "batch:SubmitJob",
            "batch:ListJobs",
            "batch:DescribeJobs"
         ],
         "Resource":"*"
      }
   ]
}
```

合规性验证 AWS Batch

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全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，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 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的最佳实践， AWS 服务并将指南映射到跨多个框架（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)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。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 — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

中的基础设施安全 AWS Batch

作为一项托管服务 AWS Batch ，受 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用 AWS Batch 通过网络进行访问。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。

您可以从任何网络位置调用这些 API 操作， AWS Batch 但支持基于资源的访问策略，其中可能包括基于源 IP 地址的限制。您还可以使用 AWS Batch 策略来控制来自特定亚马逊虚拟私有云 (Amazon VPC) 终端节点或特定 VPC 的访问。实际上，这可以将对给定 AWS Batch 资源的网络访问与网络中的特定 VPC 隔离开来。 AWS

对AWS Batch资源加标签

为了帮助管理AWS Batch资源，可通过标签的形式为每个资源分配元数据。本主题介绍标签并演示如何创建标签。

目录

有关标签的基本知识

标签是为AWS资源分配的标记。每个标签都包含定义的一个键 和一个可选值。

标签允许按用途、所有者或环境等对AWS资源进行分类。在具有相同类型的许多资源时，可以根据分配给资源的标签快速识别具体的资源。例如，可以为AWS Batch服务定义一组标签，以帮助跟踪每个服务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。

标签不会自动分配至资源。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AWS Batch没有任何语义意义，应严格按字符串进行解析。可以将标签的值设为空的字符串，但是不能将其设为空值。如果添加的标签的键与该资源上现有标签的键相同，新值就会覆盖旧值。

可以使用AWS Management Console、AWS CLI和AWS Batch API 处理标签。

如果使用的是AWS Identity and Access Management (IAM)，则可以控制AWS账户中的哪些用户拥有创建、编辑或删除标签的权限。

对资源加标签

可以对新的或现有的AWS Batch计算环境、作业、作业定义、作业队列和计划策略加标签。

如果使用的是AWS Batch控制台，则可以在创建新资源时对其应用标签，或随时在相关资源页面上使用标签选项卡对现有资源应用标签。

如果使用的是AWS Batch API、AWS CLI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，或使用`TagResource` API 操作对现有资源应用标签。有关更多信息，请参阅TagResource。

某些资源创建操作允许在创建资源时为其指定标签。如果无法在资源创建期间应用标签，资源创建过程失败。这可确保对于要在创建时加标签的资源，要么使用指定的标签创建，要么完全不创建。如果在创建时对资源加标签，则无需在资源创建后运行自定义对脚本加标签。

下表描述了可以加标签的AWS Batch资源以及可在创建时加标签的资源。

给AWS Batch资源加标签支持

资源	支持标签	支持标签传播	支持在创建时添加标签（AWS BatchAPI、AWS CLI、AWSSDK）
AWS Batch计算环境	是	不是。计算环境标签不传播到任何其他资源。资源的标签在CreateComputeEnvironment API 操作中传递的 computeResources 对象的标签成员中指定。	是
AWS Batch 个作业	是	是	是
AWS Batch 个作业定义	是	否	是
AWS Batch 个作业队列	是	否	是
AWS Batch 个计划策略	是	否	是

标签限制

下面是适用于标签的基本限制：

- 每个资源的标签数上限 – 50
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大键长度 – 128 个 Unicode 字符（采用 UTF-8 格式）
- 最大值长度 – 256 个 Unicode 字符 (采用 UTF-8 格式)
- 如果标签方案针对多个AWS服务和资源使用，请记得其它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可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：+ - = . _ : / @。
- 标签键和值区分大小写。
- 请不要使用aws:、AWS:或此类拼写的任意大小写组合作为键或值的前缀，因为它将保留以供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。

通过控制台使用标签

可以使用AWS Batch控制台管理与新的或现有的计算环境、任务、作业定义和作业队列关联的标签。

在创建时为单个资源添加标签

可以在创建AWS Batch计算环境、作业、作业定义、作业队列和计划策略时为它们添加标签。

为单个资源添加和删除标签

AWS Batch允许直接从资源的页面中添加或删除与集群相关的标签。

添加或删除单个资源上的标签

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择资源类型（例如，作业队列）。
4. 选择一项具体资源，然后选择编辑标签。
5. 根据需要添加或删除标签。
   - 添加标签 — 在列表末尾的空白文本框中指定键和值。
   - 要删除标签 — 选择标签旁边的

Delete icon
按钮。

6. 为要添加或删除的每个标签重复此过程，然后选择编辑标签以完成操作。

通过 CLI 或 API 使用标签

使用以下AWS CLI命令或AWS Batch API 操作来添加、更新、列出和删除资源的标签。

给AWS Batch资源加标签支持

任务	API 操作	AWS CLI	AWS Tools for Windows PowerShell
添加或覆盖一个或多个标签。	TagResource	tag-resource	Add-BATResourceTag
删除一个或多个标签。	UntagResource	untag-resource	Remove-BATResourceTag
列出资源的标签	ListTagsForResource	list-tags-for-resource	Get-BATResourceTag

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：对现有资源加标签

以下命令对现有资源加标签。

```
aws batch tag-resource --resource-arn resource_ARN --tags team=devs
```

示例 2：对现有资源取消标签

以下命令删除现有资源的标签。

```
aws batch 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 batch list-tags-for-resource --resource-arn resource_ARN
```

某些资源创建操作允许在创建资源时指定标签。以下操作支持在创建时加标签。

任务	API 操作	AWS CLI	AWS Tools for Windows PowerShell
创建计算环境	CreateComputeEnvironment	create-compute-environment	New-BATComputeEnvironment
创建作业队列	CreateJobQueue	create-job-queue	New-BATJobQueue
创建计划策略	CreateSchedulingPolicy	create-scheduling-policy	New-BATSchedulingPolicy
注册作业定义	RegisterJobDefinition	register-job-definition	Register-BATJobDefinition
提交作业	SubmitJob	submit-job	Submit-BATJob

AWS Batch 服务限额

下表提供了无法更改的 AWS Batch 服务限额。每个限额都是针对特定区域的。

资源	限额
最大作业队列数。有关更多信息，请参阅 作业队列。	50
Amazon ECS 和 Amazon EKS 计算环境的最大数量。有关更多信息，请参阅 计算环境。	50
每个 Amazon EKS 集群的计算环境的最大数量。	5
每个作业队列的计算环境的最大数量	3
一项作业的最大作业依赖项数	20
最大任务定义大小 (对于 `RegisterJobDefinition` API 操作)	24 KiB
最大作业负载大小（对于 `SubmitJob` API 操作）	30 KiB
数组作业的最大数组大小	10000
处于 `SUBMITTED` 状态的最大作业数	1000000
每个 `SubmitJob` 操作账户每秒 (TPS) 可处理的最大处理数	50

根据您使用 AWS Batch 的方式，可能会适用额外的限额。要了解有关 Amazon EC2 限额的信息，请参阅AWS 一般参考中的 Amazon EC2 服务限额。有关 Amazon ECS 限额的更多信息，请参阅AWS 一般参考中的 Amazon ECS 服务限额。有关 Amazon EKS 限额的更多信息，请参阅AWS 一般参考中的 Amazon EKS 服务限额。

故障排除 AWS Batch

可能需要排除与计算环境、作业队列、作业定义或作业相关的问题。本章介绍如何对您的 AWS Batch 环境中的此类问题进行故障排除和解决。

AWS Batch 使用 IAM 策略、角色和权限，在亚马逊 EC2、Amazon ECS 和亚马逊 Elastic Kubernetes Service 基础设施上运行。 AWS Fargate要解决与这些服务相关的问题，请参阅以下内容：

- 《IAM 用户指南》中的 IAM 故障排除
- Amazon Elastic Container Service 开发人员指南中的Amazon ECS 故障排除
- 《Amazon EKS 用户指南》中的 Amazon EKS 故障排除
- 在 Amazon EC2 用户指南中对 EC2 实例进行故障排除

目录

AWS Batch

INVALID 计算环境

您可能错误地配置了托管计算环境。如果是这样，计算环境就会进入INVALID状态，无法接受作业放置。以下各节描述了可能的原因以及如何根据原因进行故障排除。

角色名称或 ARN 不正确

计算环境进入INVALID状态的最常见原因是 AWS Batch 服务角色或 Amazon EC2 Spot 队列角色的名称或亚马逊资源名称 (ARN) 不正确。这在使用 AWS CLI 或 AWS 软件开发工具包创建的计算环境中更为常见。当您在中创建计算环境时 AWS Management Console， AWS Batch 可以帮助您选择正确的服务或 Spot 队列角色。但是，假设您手动输入了名称或 ARN，但输入不正确。然后，生成的计算环境也是INVALID。

但是，假设在 AWS CLI 命令或 SDK 代码中手动输入 IAM 资源的名称或 ARN。在这种情况下， AWS Batch 无法验证字符串。相反， AWS Batch 必须接受坏值并尝试创造环境。如果 AWS Batch 无法创建环境，则环境将变为INVALID状态，并且您会看到以下错误。

对于无效的服务角色：

```
CLIENT_ERROR - Not authorized to perform sts:AssumeRole (Service:
AWSSecurityTokenService; Status Code: 403; Error Code: AccessDenied;
Request ID: dc0e2d28-2e99-11e7-b372-7fcc6fb65fe7)
```

对于无效的竞价型实例集角色：

```
CLIENT_ERROR - Parameter: SpotFleetRequestConfig.IamFleetRole
is invalid. (Service: AmazonEC2; Status Code: 400; Error Code:
InvalidSpotFleetRequestConfig; Request ID: 331205f0-5ae3-4cea-
bac4-897769639f8d) Parameter: SpotFleetRequestConfig.IamFleetRole is
invalid
```

导致此问题的常见原因之一是以下情况。使用 AWS CLI 或 AWS 软件开发工具包时，您只能指定 IAM 角色的名称，而不是完整的 Amazon 资源名称 (ARN)。根据创建角色的方式，ARN 可能包含aws-service-role路径前缀。例如，如果使用将服务相关角色用于 AWS Batch中的程序手动创建 AWS Batch 服务角色，服务角色 ARN 可能如下所示。

```
arn:aws:iam::123456789012:role/AWSBatchServiceRole
```

但是，如果在控制台首次运行向导中创建了服务角色，则服务角色 ARN 可能如下所示。

```
arn:aws:iam::123456789012:role/aws-service-role/AWSBatchServiceRole
```

如果您将 AWS Batch 服务级别策略 (AWSBatchServiceRole) 附加到非服务角色，也会出现此问题。例如，在这种情况下，可能会收到类似于以下内容的错误消息：

```
CLIENT_ERROR - User: arn:aws:sts::account_number:assumed-role/batch-replacement-role/
aws-batch is not
     authorized to perform: action on resource ...
```

要解决该问题，可以执行下列操作之一：

- 创建 AWS Batch 计算环境时，请为服务角色使用空字符串。
- 采用以下格式指定服务角色：arn:aws:iam::*account_number*:role/aws-service-role/batch.amazonaws.com/AWSServiceRoleForBatch。

如果您在使用 AWS CLI 或 AWS 软件开发工具包时仅指定 IAM 角色的名称，则 AWS Batch 假设您的 ARN 不使用路径aws-service-role前缀。因此，我们建议在创建计算环境时为 IAM 角色指定完整 ARN。

要修复以这种方式错误配置的计算环境，请参阅修复INVALID计算环境。

修复INVALID计算环境

当拥有处于INVALID状态的计算环境时，请更新它以修复无效参数。对于角色名称或 ARN 不正确，可以使用正确的服务角色更新计算环境。

修复配置错误的计算环境

1. 打开 AWS Batch 控制台，网址为 https://console.aws.amazon.com/batch/。
2. 在导航栏中，选择 AWS 区域 要使用的。

3. 在导航窗格中，选择计算环境。
4. 在计算环境页面上，选择要编辑的计算环境旁边的单选按钮，然后选择编辑。
5. 在更新计算环境页面上，对于服务角色，请选择要用于计算环境的 IAM 角色。 AWS Batch 控制台仅显示与计算环境具有正确信任关系的角色。
6. 选择保存以更新计算环境。

作业在RUNNABLE状态卡住

假设计算环境包含计算资源，但作业不会超出RUNNABLE状态。然后，很可能是某些因素阻止了将作业放在计算资源上，并导致您的任务队列被阻止。以下是如何知道你的工作是在等待轮到它还是卡住并阻塞队列。

如果 AWS Batch 检测到您的头部有RUNNABLE任务并阻止了队列，您将收到来自 Amazon Events 的已阻止任务队列 CloudWatch 事件，并附有原因。同样的原因也作为ListJobs和 DescribeJobs API 调用的一部分更新到该statusReason字段。

或者，您可以通过CreateJobQueue和 UpdateJobQueueAPI 操作配置jobStateTimeLimitActions参数。

> **Note**
>
> 目前，您可以使用的唯一操作jobStateLimitActions.action是取消任务。

该jobStateTimeLimitActions参数用于指定对处于特定状态的作业 AWS Batch 执行的一组操作。您可以通过该maxTimeSeconds字段设置以秒为单位的时间阈值。

当作业处于已定义的RUNNABLE状态时statusReason，将在经过后 AWS Batch maxTimeSeconds执行指定的操作。

例如，对于处于等待足够容量可用RUNNABLE状态的任何作业，您可以将jobStateTimeLimitActions参数设置为最多等待 4 小时。为此，您可以先将该任务设置statusReasonmaxTimeSeconds为CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY和设置为144000，然后再取消任务并允许下一个作业进入任务队列的开头。

以下是它检测到作业队列被阻塞时 AWS Batch 提供的原因。此列表提供了从ListJobs和 DescribeJobs API 操作返回的消息。这些值也与您可以为jobStateLimitActions.statusReason参数定义的值相同。

1. 原因：所有连接的计算环境都存在容量不足错误。在收到请求时， AWS Batch 会检测出现容量不足错误的 Amazon EC2 实例。手动或通过将`jobStateTimeLimitActions`参数设置为开启来取消作业`statusReason`，可以将后续作业移到队列的开头。

   - **`statusReason`**任务卡住时发消息：`CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY - Service cannot fulfill the capacity requested for instance type [instanceTypeName]`
   - **`reason`**用于**`jobStateTimeLimitActions`**：`CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY`
   - **`statusReason`**任务取消后的消息：`Canceled by JobStateTimeLimit action due to reason: CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY`

   注意：

   a. AWS Batch 服务角色需要`autoscaling:DescribeScalingActivities`权限才能使此检测生效。如果您使用AWSServiceRoleForBatch服务相关角色 (SLR) 或AWSBatchServiceRolePolicy托管策略，则无需采取任何操作，因为他们的权限策略已更新。

   b. 如果您使用 SLR 或托管策略，则必须添加`autoscaling:DescribeScalingActivities`和`ec2:DescribeSpotFleetRequestHistory`权限，以便在进入时可以接收已阻止的作业队列事件和更新的作业状态。`RUNNABLE`此外， AWS Batch 需要这些权限才能通过`jobStateTimeLimitActions`参数执行`cancellation`操作，即使它们是在作业队列中配置的。

   c. 对于多节点并行 (MNP) 作业，如果附加的高优先级 Amazon EC2 计算环境遇到`insufficient capacity`错误，即使优先级较低的计算环境确实遇到此错误，它也会阻塞队列。

2. 原因：所有计算环境的maxvCpus参数都小于任务要求。手动或通过将`jobStateTimeLimitActions`参数设置为开启来取消作业`statusReason`，可以将后续作业移到队列的开头。或者，您可以增加主计算环境的`maxvCpus`参数以满足受阻作业的需求。

   - **`statusReason`**任务卡住时发消息：`MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE - CE(s) associated with the job queue cannot meet the CPU requirement of the job.`
   - **`reason`**用于**`jobStateTimeLimitActions`**：`MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE`
   - **`statusReason`**任务取消后的消息：`Canceled by JobStateTimeLimit action due to reason: MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE`

3. 原因：所有计算环境都没有满足任务要求的实例。当任务请求资源时，AWS Batch 会检测到没有连接的计算环境能够容纳传入的作业。手动或通过将jobStateTimeLimitActions参数设置为开启来取消作业statusReason，可以将后续作业移到队列的开头。或者，您可以重新定义计算环境允许的实例类型，以添加必要的作业资源。

   - **statusReason**任务卡住时发消息： `MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT - The job resource requirement (vCPU/memory/GPU) is higher than that can be met by the CE(s) attached to the job queue.`
   - **reason**用于**jobStateTimeLimitActions**：`MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT`
   - **statusReason**任务取消后的消息：`Canceled by JobStateTimeLimit action due to reason: MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT`

4. 原因：所有计算环境都存在服务角色问题。要解决此问题，请将您的服务角色权限与AWS Batch 托管服务角色权限进行比较并填补任何差距。

   最佳做法是在计算环境中使用AWS Batch SLR 以避免类似错误。

   手动或通过将jobStateTimeLimitActions参数设置为开启来取消作业statusReason，可以将后续作业移到队列的开头。如果不解决服务角色问题，下一个工作也可能被阻止。最好手动调查并解决此问题。

   - **statusReason**任务卡住时发消息：`MISCONFIGURATION:SERVICE_ROLE_PERMISSIONS - Batch service role has a permission issue.`
   - **reason**用于**jobStateTimeLimitActions**：`MISCONFIGURATION:SERVICE_ROLE_PERMISSIONS`
   - **statusReason**任务取消后的消息：`Canceled by JobStateTimeLimit action due to reason: MISCONFIGURATION:SERVICE_ROLE_PERMISSIONS`

5. 原因：所有计算环境均无效。有关更多信息，请参阅INVALID计算环境。注意：您无法通过jobStateTimeLimitActions参数配置可编程操作来解决此错误。

   - **statusReason**任务卡住时发消息：`ACTION_REQUIRED - CE(s) associated with the job queue are invalid.`

6. 原因： AWS Batch 已检测到队列被阻塞，但无法确定原因。注意：您无法通过jobStateTimeLimitActions参数配置可编程操作来解决此错误。有关疑难解答的更多信息，请参阅 re: Post 中为什么我的 AWS Batch 工作停留在 RUNNABLE 开启 AWS。

   - **statusReason**任务卡住时发消息：`UNDETERMINED - Batch job is blocked, root cause is undetermined.`

如果您没有收到来自事件 CloudWatch 的事件或收到未知原因事件，则以下是导致此问题的一些常见原因。

未在您的计算资源上配置**awslogs**日志驱动程序

AWS Batch 作业将其日志信息发送到 CloudWatch 日志。为了做到这一点，必须将计算资源配置为使用awslogs日志驱动程序。假设计算资源 AMI 基于 Amazon ECS 优化 AMI（或 Amazon Linux）。然后，该驱动程序默认会注册到ecs-init软件包中。假设现在使用的是不同的基础 AMI。然后，必须验证在启动 Amazon ECS 容器代理时，是否使用ECS_AVAILABLE_LOGGING_DRIVERS环境变量将awslogs日志驱动程序指定为可用的日志驱动程序。有关更多信息，请参阅计算资源 &AMI; 规范和创建计算资源 &AMI;：

资源不足

如果作业定义指定的 CPU 或内存资源超出可分配的计算资源量，则作业将永远不会被放置。例如，假设作业指定了 4 GiB 的内存，而计算资源少于可用内存。那么，作业就不能放置在这些计算资源上。在这种情况下，必须减少作业定义中指定的内存，或者向环境中添加更大的计算资源。为 Amazon ECS 容器代理和其他关键系统进程保留了部分内存。有关更多信息，请参阅 计算资源内存管理。

计算资源无法访问互联网

计算资源需要访问才能与 Amazon ECS 服务端点通信。这可以通过接口 VPC 端点或具有公共 IP 地址的计算资源实现。

有关接口 VPC 端点的更多信息，请参阅 Amazon Elastic Container Service 开发人员指南中的 Amazon ECS 接口 VPC 端点(AWS PrivateLink)。

如果没有配置接口 VPC 端点，并且计算资源没有公共 IP 地址，则必须使用网络地址转换 (NAT) 来提供这种访问。有关更多信息，请参阅《Amazon VPC 用户指南》中的。???有关更多信息，请参阅 the section called “创建 VPC”。

已达到亚马逊 EC2 实例限制

您的账户可以在中启动的 Amazon EC2 实例数量由您 AWS 区域 的 EC2 实例配额决定。某些实例类型也有配 per-instance-type额。有关您账户的 Amazon EC2 实例配额的更多信息，包括如何申请提高限额，请参阅 Amazon EC2 用户指南中的 Amazon EC2 服务限制。

未安装 Amazon ECS 容器代理

必须将 Amazon ECS 容器代理安装在亚马逊机器映像（AMI）上才能使 AWS Batch 运行作业。Amazon ECS 容器代理默认安装在 Amazon ECS 优化 AMI 上。有关 Amazon ECS 容器代理

的更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 容器代理。

如需了解更多信息，请参阅为什么我的 AWS Batch 工作RUNNABLE状态停滞不前？ 在 re: post 中。

创建时未标记的竞价型实例

自 2017 年 10 月 25 日起，支持 AWS Batch 计算资源的竞价型实例标记。以前，Amazon EC2 竞价型实例集角色的推荐 IAM 托管策略 (AmazonEC2SpotFleetRole) 不包含在启动时标记竞价型实例的权限。推荐使用的新 IAM 托管策略称为AmazonEC2SpotFleetTaggingRole。它支持在启动时标记竞价型实例。

要修复创建竞价型实例时的标记问题，请按照以下步骤将当前推荐的 IAM 托管策略应用到 Amazon EC2 竞价型实例集角色。这样，今后使用该角色创建的任何竞价型实例在创建时都有权限应用实例标签。

要将当前 IAM 托管策略应用于 Amazon EC2 竞价型实例集角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择角色，然后选择 Amazon EC2 竞价型实例集角色。
3. 选择附上策略。
4. 选择 AmazonEC2 SpotFleet TaggingRole 并选择附加策略。
5. 再次选择 Amazon EC2 竞价型实例集角色，以移除以前的策略。
6. 选择 AmazonEC2 SpotFleet 角色策略右侧的 x，然后选择 “分离”。

竞价型实例无法缩减

AWS Batch 2021 年 3 月 10 日推出了AWSServiceRoleForBatch与服务相关的角色。如果在计算环境的serviceRole参数中未指定任何角色，则此服务相关角色将用作服务角色。但是，假设在 EC2 竞价计算环境中使用服务相关角色，但使用的竞价角色不包括 AmazonEC2 SpotFleet TaggingRole 托管策略。这样，竞价型实例就不会缩减。因此，您将收到一条错误信息，内容如下：“您无权执行此操作”。使用以下步骤更新spotIamFleetRole参数中使用的竞价型实例集角色。有关更多信息，请参阅 IAM 用户指南中的使用服务相关角色和创建角色向 AWS 服务委派权限。

主题

- 将 AmazonEC2 SpotFleet TaggingRole 托管策略附加到您的 Spot 队列角色中 AWS Management Console

- 将 AmazonEC2 SpotFleet TaggingRole 托管策略附加到您的 Spot 队列角色中 AWS CLI

将 AmazonEC2 SpotFleet TaggingRole 托管策略附加到您的 Spot 队列角色中 AWS Management Console

要将当前 IAM 托管策略应用于 Amazon EC2 竞价型实例集角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择角色，然后选择 Amazon EC2 竞价型实例集角色。
3. 选择附上策略。
4. 选择 AmazonEC2 SpotFleet TaggingRole 并选择附加策略。
5. 再次选择 Amazon EC2 竞价型实例集角色，以移除以前的策略。
6. 选择 AmazonEC2 SpotFleet 角色策略右侧的 x，然后选择 “分离”。

将 AmazonEC2 SpotFleet TaggingRole 托管策略附加到您的 Spot 队列角色中 AWS CLI

示例命令假设您的 Amazon EC2 竞价队列角色名为 *AmazonEC2 SpotFleet ##*。如果角色使用不同的名称，请调整命令以使其匹配。

将 AmazonEC2 SpotFleet TaggingRole 托管策略附加到您的 Spot 队列角色

1. 要将 AmazonEC2 SpotFleet TaggingRole 托管 IAM 策略附加到您*# AmazonEC2 ##SpotFleet ##*，请使用运行以下命令。 AWS CLI

```
$ aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2SpotFleetTaggingRole
 \
    --role-name AmazonEC2SpotFleetRole
```

2. 要将 AmazonEC2 SpotFleet 角色托管 IAM 策略与您的 A*mazonEC2 SpotFleet ##*角色分离，请使用运行以下命令。 AWS CLI

```
$ aws iam detach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2SpotFleetRole \
    --role-name AmazonEC2SpotFleetRole
```

无法检索 Secrets Manager 密文

如果将 AMI 与早于 1.16.0-1 版本的 Amazon ECS 代理一起使用，则必须使用 Amazon ECS 代理配置变量`ECS_ENABLE_AWSLOGS_EXECUTIONROLE_OVERRIDE=true`才能使用此功能。可以在创建新容器实例时将其添加到该实例的`./etc/ecs/ecs.config`文件中。或者，可以将其添加到现有实例。如果将其添加到现有实例中，则必须在添加 ECS 代理后重新启动 ECS 代理。有关更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 容器代理配置。

无法覆盖作业定义资源需求

在传递给的 C ontainerOverrides 结构`memory`和`vcpus`成员中指定的内存和 vCPU 覆盖不能覆盖任务定义的 ResourceRequiredes 结构中指定的内存和 vCPU 要求。SubmitJob

如果尝试覆盖这些资源需求，可能会出现以下错误消息：

“此值是在已弃用的密钥中提交的，可能与作业定义的资源需求提供的值冲突。”

要更正此问题，请在 containerOverrides 的 resourceRequirements 成员中指定内存和 vCPU 需求。例如，如果在以下行中指定了内存和 vCPU 替代项。

```
"containerOverrides": {
   "memory": 8192,
   "vcpus": 4
}
```

将其更改为以下内容：

```
"containerOverrides": {
   "resourceRequirements": [
      {
         "type": "MEMORY",
         "value": "8192"
      },
      {
         "type": "VCPU",
         "value": "4"
      }
   ],
}
```

对作业定义的 containerProperties 对象中指定的内存和 vCPU 需求进行相同的更改。例如，如果在以下几行中指定了内存和 vCPU 需求。

```
{
   "containerProperties": {
      "memory": 4096,
      "vcpus": 2,
}
```

将其更改为以下内容：

```
"containerProperties": {
   "resourceRequirements": [
      {
         "type": "MEMORY",
         "value": "4096"
      },
      {
         "type": "VCPU",
         "value": "2"
      }
   ],
}
```

更新desiredvCpus设置时出现错误消息

当您使用 AWS Batch API 更新所需的 vCPU desiredvCpus () 设置时，您会看到以下错误消息。

```
Manually scaling down compute environment is not supported. Disconnecting job queues from compute environment will cause it to scale-down to minvCpus.
```

如果更新的desiredvCpus值小于当前desiredvCpus值，则会出现此问题。更新desiredvCpus值时，必须满足以下两个条件：

- desiredvCpus值必须介于minvCpus值和maxvCpus值之间。
- 更新的desiredvCpus值必须大于或等于当前的desiredvCpus值。

AWS Batch 在亚马逊 EKS 上

主题

INVALID 计算环境

您可能错误地配置了托管计算环境。如果是这样，计算环境就会进入INVALID状态，无法接受作业放置。以下各节描述了可能的原因以及如何根据原因进行故障排除。

不支持的Kubernetes版本

当使用 CreateComputeEnvironment API 操作或 UpdateComputeEnvironmentAPI 操作以创建或更新计算环境时，可能会看到类似于以下内容的错误消息。如果在EC2Configuration中指定不受支持的Kubernetes版本，则会出现此问题。

```
At least one imageKubernetesVersion in EC2Configuration is not supported.
```

要解决此问题，请删除计算环境，然后使用支持的Kubernetes版本重新创建。

可以在 Amazon EKS 集群上执行次要版本升级。例如，即使不支持次要版本，也可以将集群从1.xx升级到1.yy。

但是，主要版本更新后，计算环境的状态可能会更改为INVALID。例如，如果将主要版本从1.xx升级到2.yy。如果不支持主要版本 AWS Batch，则会看到类似于以下内容的错误消息。

```
reason=CLIENT_ERROR - ... EKS Cluster version [2.yy] is unsupported
```

要解决此问题，请在使用 API 操作创建或更新计算环境时指定支持的Kubernetes版本。

AWS Batch 在 Amazon 上，EKS 目前支持以下Kubernetes版本：

- 1.29

- 1.28
- 1.27
- 1.26
- 1.25
- 1.24
- 1.23

实例配置文件不存在

如果指定的实例配置文件不存在，则 Amazon EKS AWS Batch 上的计算环境状态将更改为INVALID。在statusReason参数中会出现类似于以下内容的错误集。

```
CLIENT_ERROR - Instance profile arn:aws:iam::...:instance-profile/<name> does not exist
```

要解决此问题，请指定或创建有效的实例配置文件。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 节点 IAM 角色。

Kubernetes命名空间无效

如果 AWS Batch 在 Amazon 上，EKS 无法验证计算环境的命名空间，则计算环境的状态将更改为INVALID。例如，如果命名空间不存在，则可能会出现此问题。

在statusReason参数中会出现类似于以下内容的错误消息集。

```
CLIENT_ERROR - Unable to validate Kubernetes Namespace
```

在满足以下任一条件时，可能出现此问题：

- CreateComputeEnvironment调用中的Kubernetes命名空间字符串不存在。有关更多信息，请参阅CreateCompute环境。
- 管理命名空间所需的基于角色的访问控制 (RBAC) 权限配置不正确。
- AWS Batch 无法访问 Amazon EKS Kubernetes API 服务器终端节点。

要解决此问题，请参阅验证aws-auth ConfigMap是否配置正确。。有关更多信息，请参阅亚马逊 EK AWS Batch S 入门。

已删除的计算环境

假设您在删除 Amazon EKS 计算环境 AWS Batch 上连接的集群之前删除了 Amazon EKS 集群。然后，计算环境状态更改为INVALID。在这种情况下，如果使用相同的名称重新创建 Amazon EKS 集群，则计算环境将无法正常运行。

要解决此问题，请删除 Amazon EKS AWS Batch 上的计算环境，然后重新创建。

节点未加入 Amazon EKS 集群

AWS Batch 在 Amazon EKS 上，如果计算环境确定并非所有节点都加入了 Amazon EKS 集群，则会缩小计算环境的规模。 AWS Batch 在 Amazon EKS 上缩小计算环境时，计算环境的状态将更改为INVALID。

> **Note**
>
> AWS Batch 不会立即更改计算环境状态以便您可以调试问题。

在statusReason参数中会出现类似于以下内容的错误消息集：

```
Your compute environment has been INVALIDATED and scaled down because
none of the instances joined the underlying ECS Cluster. Common issues
preventing instances joining are the following: VPC/Subnet configuration
preventing communication to ECS, incorrect Instance Profile policy
preventing authorization to ECS, or customized AMI or LaunchTemplate
configurations affecting ECS agent.
```

```
Your compute environment has been INVALIDATED and scaled down because
none of the nodes joined the underlying Amazon EKS Cluster. Common issues
preventing nodes joining are the following: networking configuration
preventing communication to Amazon EKS Cluster, incorrect Amazon EKS
Instance Profile or Kubernetes RBAC policy preventing authorization
to Amazon EKS Cluster, customized AMI or LaunchTemplate configurations
affecting Amazon EKS/Kubernetes node bootstrap.
```

使用默认 Amazon EKS AMI 时，导致此问题的最常见原因如下：

- 实例角色配置不正确。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 节点 IAM 角色。

- 子网配置不正确。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS VPC 和子网要求和注意事项。
- 安全组配置不正确。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 安全组要求和注意事项。

> **Note**
>
> 在 Personal Health Dashboard (PHD) 中也可能会出现错误通知。

AWS Batch 在 Amazon 上，EKS 的工作处于**RUNNABLE**状态不变

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。最初创建的aws-authConfigMap目的是允许节点加入集群。但是，也可以使用aws-authConfigMap为用户和角色添加基于角色的访问控制 (RBAC)。

验证aws-auth ConfigMap是否配置正确。

1. 检索aws-authConfigMap中的映射角色：

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

2. 验证roleARN是否按以下方式配置。

```
rolearn: arn:aws:iam::aws_account_number:role/AWSServiceRoleForBatch
```

> **Note**
>
> 还可以查看 Amazon EKS 控制面板日志。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 控制面板日志。

要解决作业停留在RUNNABLE状态的问题，建议使用kubectl重新应用清单。有关更多信息，请参阅第 1 步：为您的 Amazon EKS 集群做好准备 AWS Batch。或者，可以kubectl使用手动编辑aws-authConfigMap。有关更多信息，请参阅《Amazon EKS 用户指南》中的启用 IAM 用户和您的集群的角色访问权限。

验证aws-auth ConfigMap是否配置正确。

验证aws-auth ConfigMap是否配置正确。

1. 检索aws-auth ConfigMap中的映射角色。

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

2. 验证roleARN是否按以下方式配置。

```
rolearn: arn:aws:iam::aws_account_number:role/AWSServiceRoleForBatch
```

> **Note**
>
> 已从服务相关角色的 ARN 中删除路径aws-service-role/batch.amazonaws.com/。这是因为 aws-auth 配置映射存在问题。有关更多信息，请参阅aws-authconfigmap中带路径的角色在其 ARN 中包含路径时不起作用。

> **Note**
>
> 还可以查看 Amazon EKS 控制面板日志。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 控制面板日志。

要解决作业停留在RUNNABLE状态的问题，建议使用kubectl重新应用清单。有关更多信息，请参阅第 1 步：为您的 Amazon EKS 集群做好准备 AWS Batch。或者，可以kubectl使用手动编辑aws-authConfigMap。有关更多信息，请参阅《Amazon EKS 用户指南》中的启用 IAM 用户和您的集群的角色访问权限。

RBAC 权限或绑定配置不正确

如果遇到任何 RBAC 权限或绑定问题，请验证aws-batchKubernetes角色是否可以访问Kubernetes命名空间：

```
$ kubectl get namespace namespace --as=aws-batch
```

```
$ kubectl auth can-i get ns --as=aws-batch
```

还可以使用**kubectl describe**命令查看集群角色或Kubernetes命名空间的授权。

```
$ kubectl describe clusterrole aws-batch-cluster-role
```

下面是示例输出。

```
Name:         aws-batch-cluster-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources                                       Non-Resource URLs  Resource Names
 Verbs
  ---------                                       -----------------  --------------
 -----
  configmaps                                      []                 []
 [get list watch]
  nodes                                           []                 []
 [get list watch]
  pods                                            []                 []
 [get list watch]
  daemonsets.apps                                 []                 []
 [get list watch]
  deployments.apps                                []                 []
 [get list watch]
  replicasets.apps                                []                 []
 [get list watch]
  statefulsets.apps                               []                 []
 [get list watch]
  clusterrolebindings.rbac.authorization.k8s.io   []                 []
 [get list]
  clusterroles.rbac.authorization.k8s.io          []                 []
 [get list]
  namespaces                                      []                 []
 [get]
```

```
$ kubectl describe role aws-batch-compute-environment-role -n my-aws-batch-namespace
```

下面是示例输出。

```
Name:         aws-batch-compute-environment-role
Labels:       <none>
Annotations:  <none>
```

```
PolicyRule:
  Resources                                  Non-Resource URLs  Resource Names  Verbs
  ---------                                  -----------------  --------------  -----
  pods                                       []                 []              [create
 get list watch delete patch]
  serviceaccounts                            []                 []              [get list]
  rolebindings.rbac.authorization.k8s.io     []                 []              [get list]
  roles.rbac.authorization.k8s.io            []                 []              [get list]
```

要解决此问题，请重新应用 RBAC 权限和`rolebinding`命令。有关更多信息，请参阅 第 1 步：为您的 Amazon EKS 集群做好准备 AWS Batch。

AWS Batch 的最佳实践

您可以使用 AWS Batch 来大规模运行各种要求苛刻的计算工作负载，而无需管理复杂的架构。AWS Batch 作业可广泛用于流行病学、游戏和机器学习等领域的各种用例。

本主题涵盖使用 AWS Batch 时应考虑的最佳实践，以及使用 AWS Batch 时如何运行和优化工作负载的指导。

主题

何时使用 AWS Batch

AWS Batch 以低成本大规模运行作业，并提供排队服务和成本优化的扩展。但是，并非所有工作负载都适合使用 AWS Batch 运行。

- 短作业 - 如果作业仅运行几秒钟，则调度批处理作业所需的开销可能比作业本身的运行时更长。解决方法是，在 AWS Batch 中提交任务之前，请将任务一起 binpack。然后，将您的 AWS Batch 作业配置为迭代任务。例如，将单个任务参数暂存到 Amazon DynamoDB 表或作为文件存入 Amazon S3 存储桶。考虑对任务进行分组，使每个作业运行 3-5 分钟。在 binpack 作业后，在 AWS Batch 作业中循环遍历任务组。
- 必须立即运行的作业AWS Batch – 可以快速处理作业。但是，AWS Batch 是一个调度程序，可针对性价比、作业优先级和吞吐量进行优化。AWS Batch 可能需要时间来处理您的请求。如果您需要在几秒钟内得到响应，那么使用 Amazon ECS 或 Amazon EKS 的基于服务的方法更合适。

大规模运行核对清单

在 5 万或更多 vCPU 上运行大型工作负载之前，请考虑以下核对清单。

> **Note**
>
> 如果您计划在一百万或更多 vCPU 上运行大量工作负载，或者需要大规模运行的指导，请联系您的 AWS 团队。

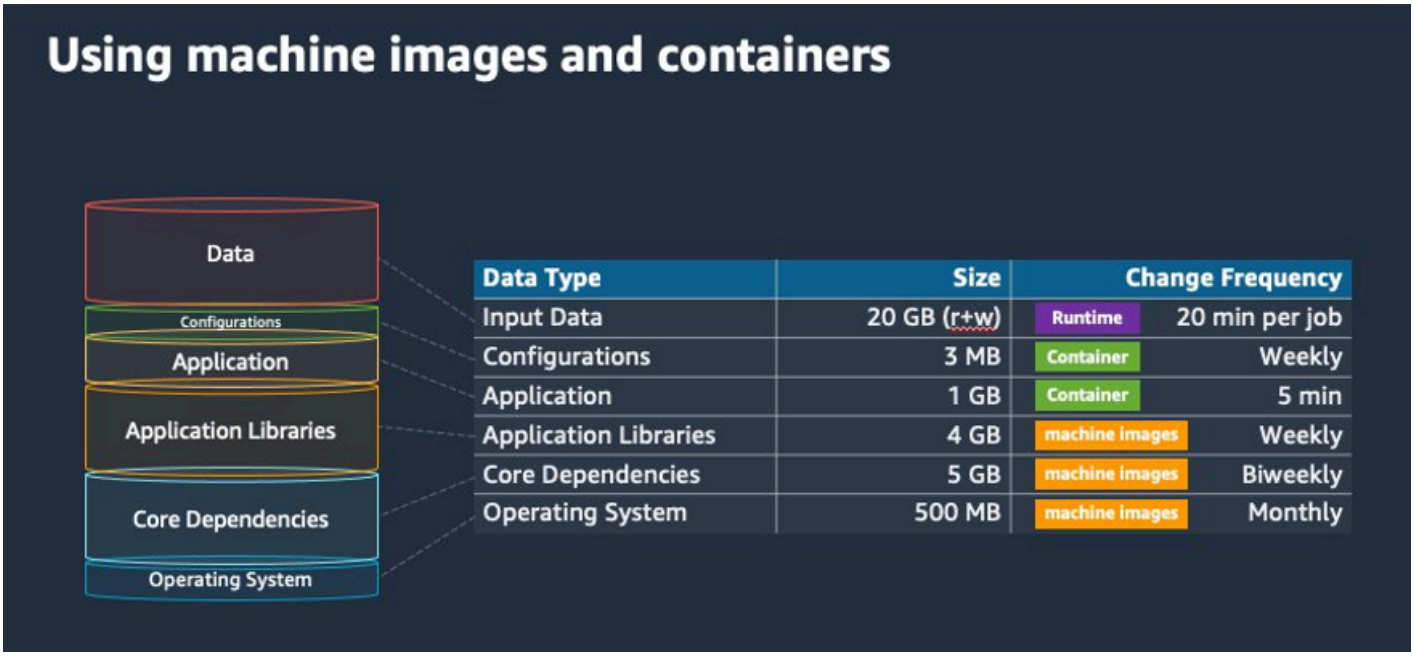
- 查看您的 Amazon EC2 限额 – 在 AWS Management Console 的“服务限额”面板中查看您的 Amazon EC2 限额（也称为限制）。如有必要，可以申请增加您的 Amazon EC2 实例峰值数量的限额。请记住，Amazon EC2 竞价型和 Amazon 按需型实例有单独的限额。有关更多信息，请参阅服务限额入门。
- 验证每个区域的 Amazon Elastic Block Store 限额 – 每个实例对操作系统使用 GP2 或 GP3 卷。默认情况下，每个 AWS 区域 限额为 300 TiB。但是，每个实例都使用计数作为此限额的一部分。因此，在验证每个区域的 Amazon Elastic Block Store 限额时，请务必将其考虑在内。如果达到限额，则无法创建更多实例。有关更多信息，请参阅 Amazon Elastic Block Store 端点和限额
- 使用 Amazon S3 进行存储 – Amazon S3 提供高吞吐量，有助于消除根据每个可用区域中的作业和实例数量来猜测要配置多少存储空间。有关更多信息，请参阅最佳实践设计模式：优化 Amazon S3 性能。
- 逐步扩展以尽早发现瓶颈 – 对于在一百万或更多 vCPU 上运行的作业，从较低的起点开始并逐渐增加，这样您就可以尽早发现瓶颈。例如，首先在 5 万个 vCPU 上运行。然后，将计数增加到 20 万 vCPU，然后增加到 50 万 vCPU，依此类推。换句话说，继续逐渐增加 vCPU 数量，直到达到所需的 vCPU 数量。
- 监控以尽早发现潜在问题 – 为了避免在大规模运行时出现潜在的中断和问题，请务必同时监控您的应用程序和架构。即使从 1 千个 vCPU 扩展到 5 千个 vCPU，也可能会出现中断。您可以使用 Amazon CloudWatch Logs 来查看日志数据，也可以使用客户端库使用 CloudWatch 嵌入式指标。有关更多信息，请参阅 CloudWatch Logs 代理参考和`aws-embedded-metrics`

优化容器和 AMI

容器大小和结构对于您运行的第一组作业非常重要。如果容器大于 4 GB，则尤其如此。容器映像是分层构建的。Docker 使用三个并发线程并行检索层。您可以使用 `max-concurrent-downloads` 参数增加并发线程数。有关更多信息，请参阅 Docker 文档。

尽管您可以使用更大的容器，但我们建议您优化容器的结构和大小，以缩短启动时间。

- 较小的容器可以更快地获取 – 较小的容器可以缩短应用程序的启动时间。要减小容器大小，请将不经常更新的库或文件卸载到亚马逊机器映像（AMI）。您也可以使用绑定挂载，为容器授予访问权限。有关更多信息，请参阅 绑定挂载。
- 创建大小均匀的层并分解大型层 — 每个层都由一个线程检索。因此，较大的层可能会显著影响您的作业启动时间。我们建议最大层大小为 2 GB，以便在更大的容器大小和更快的启动时间之间进行权衡。您可以运行 `docker history your_image_id` 命令来检查您的容器映像结构和层大小。有关更多信息，请参阅 Docker 文档。
- 使用 Amazon Elastic Container Registry 作为您的容器存储库 – 当您并行运行数千个作业时，自我管理的存储库可能会失败或限制吞吐量。Amazon ECR 可以大规模运行，可以处理多达一百多万个 vCPU 的工作负载。



选择正确的计算环境资源

与 Amazon EC2 相比，AWS Fargate 所需的初始设置和配置更少，而且可能更易于使用，尤其是在您首次使用时。使用 Fargate，您无需管理服务器、处理容量计划或隔离容器工作负载即可获得安全。

如果您有以下要求，我们建议您使用 Fargate 实例：

- 作业必须快速启动，具体时间少于 30 秒。
- 您的作业要求为 16 个 vCPU 或更少、没有 GPU 和 120 GiB 或更少的内存。

有关更多信息，请参阅 何时使用 Fargate。

如果您有以下要求，我们建议您使用 Amazon EC2 实例：

- 您需要加强对实例选择的控制，或者需要使用特定的实例类型。
- 您的作业需要 AWS Fargate 无法提供的资源，例如 GPU、更多内存、自定义 AMI 或 Amazon Elastic Fabric 适配器。
- 您需要较高的吞吐量或并发度。
- 您需要自定义 AMI、Amazon EC2 启动模板或访问特殊的 Linux 参数。

借助 Amazon EC2，您可以根据自己的特定要求更精细地调整工作负载，并在需要时大规模运行。

Amazon EC2 按需型或 Amazon EC2 竞价型

大多数 AWS Batch 客户之所以使用 Amazon EC2 竞价型实例，是因为与按需型实例相比，可以节省开支。但是，如果您的工作负载运行多个小时且无法中断，则按需型实例可能更适合您。您可以随时先试用竞价型实例，必要时切换到按需型实例。

如果您有以下要求和期望，请使用 Amazon EC2 按需型实例：

- 作业的运行时超过一个小时，您不能容忍工作负载中断。
- 您的总体工作负载有严格的 SLO（服务级别目标），并且不能增加计算时间。
- 您需要的实例更有可能出现中断。

如果您有以下要求和期望，请使用 Amazon EC2 竞价型实例：

- 作业的运行时通常为 30 分钟或更短。
- 您可以容忍潜在的中断，以及作业重新安排作为工作负载的一部分。有关更多信息，请参阅竞价型实例。
- 如果中断，可以从检查点重新启动长时间运行的作业。

您可以混合使用两种购买模式，方法是先在竞价型实例上提交，然后使用按需型实例作为后备选项。例如，在与 Amazon EC2 竞价型实例上运行的计算环境相连的队列上提交您的作业。如果作业被中断，请从 Amazon EventBridge 中捕获该事件，并将其与竞价型实例回收关联起来。然后，使用 AWS Lambda 函数或 AWS Step Functions 将作业重新提交到按需队列。有关更多信息，请参阅 教程：针对作业失败事件发送 Amazon Simple Notification Service 警报，处理 Amazon EC2 竞价型实例中断的最佳实践和使用 Step Functions 管理AWS Batch。

> **Important**
>
> 为您的按需计算环境使用不同的实例类型、大小和可用区，以维持 Amazon EC2 竞价型实例池的可用性并降低中断率。

使用 Amazon EC2 Spot 最佳实践用于 AWS Batch

当您选择 Amazon Elastic Compute Cloud (EC2) 竞价型实例时，您可能可以优化工作流程以节省成本，有时甚至可以显著节省成本。有关更多信息，请参阅 Amazon EC2 Spot 的最佳实践。

要优化您的工作流程以节省成本，请考虑以下 AWS Batch 的 Amazon EC2 Spot 最佳实践：

- 选择 **SPOT_CAPACITY_OPTIMIZED** 分配策略 – AWS Batch 从最深的 Amazon EC2 Spot 容量池中选择 Amazon EC2 实例。如果您担心中断，这是一个合适的选择。有关更多信息，请参阅 分配策略。
- 多样化实例类型 – 要使您的实例类型多样化，请考虑兼容的大小和系列，然后根据价格或可用性进行 AWS Batch 选择。例如，考虑将 `c5.24xlarge` 作为 `c5.12xlarge` 或 `c5a`、`c5n`、`c5d`、`m5` 和 `m5d` 系列的替代方案。有关更多信息，请参阅灵活选择实例类型和可用区。
- 减少作业运行时或检查点 – 我们建议不要在使用 Amazon EC2 竞价型实例运行需要一小时或更长时间的作业，以免中断。如果将作业分成小部分或设置检查点，时间不超过 30 分钟，则可以显著降低中断的可能性。
- 使用自动重试 – 为避免 AWS Batch 作业中断，请为作业设置自动重试。批处理作业可能由于以下任何原因而中断：返回非零的退出代码、发生服务错误或发生实例回收。您最多可以设置 10 次自动重试。首先，我们建议您至少设置 1-3 自动重试。有关跟踪 Amazon EC2 Spot 中断的信息，请参阅 Spot 中断控制面板。

  对于 AWS Batch，如果您设置了重试参数，则作业将放在作业队列的前面。也就是说，作业被优先考虑。在 AWS CLI 中创建作业定义或提交作业时，可以配置重试策略。有关更多信息，请参阅 提交任务。

```
$ aws batch submit-job --job-name MyJob \
    --job-queue MyJQ \
    --job-definition MyJD \
    --retry-strategy attempts=2
```

- 使用自定义重试次数 – 您可以将作业重试策略配置为特定的应用程序退出代码或实例回收。在以下示例中，如果主机导致故障，则作业最多可以重试五次。但是，如果作业由于其他原因失败，则作业将退出并将状态设置为 FAILED。

```
"retryStrategy": {
    "attempts": 5,
    "evaluateOnExit":
    [{
        "onStatusReason" :"Host EC2*",
        "action": "RETRY"
    },{
      "onReason" : "*"
        "action": "EXIT"
    }]
}
```

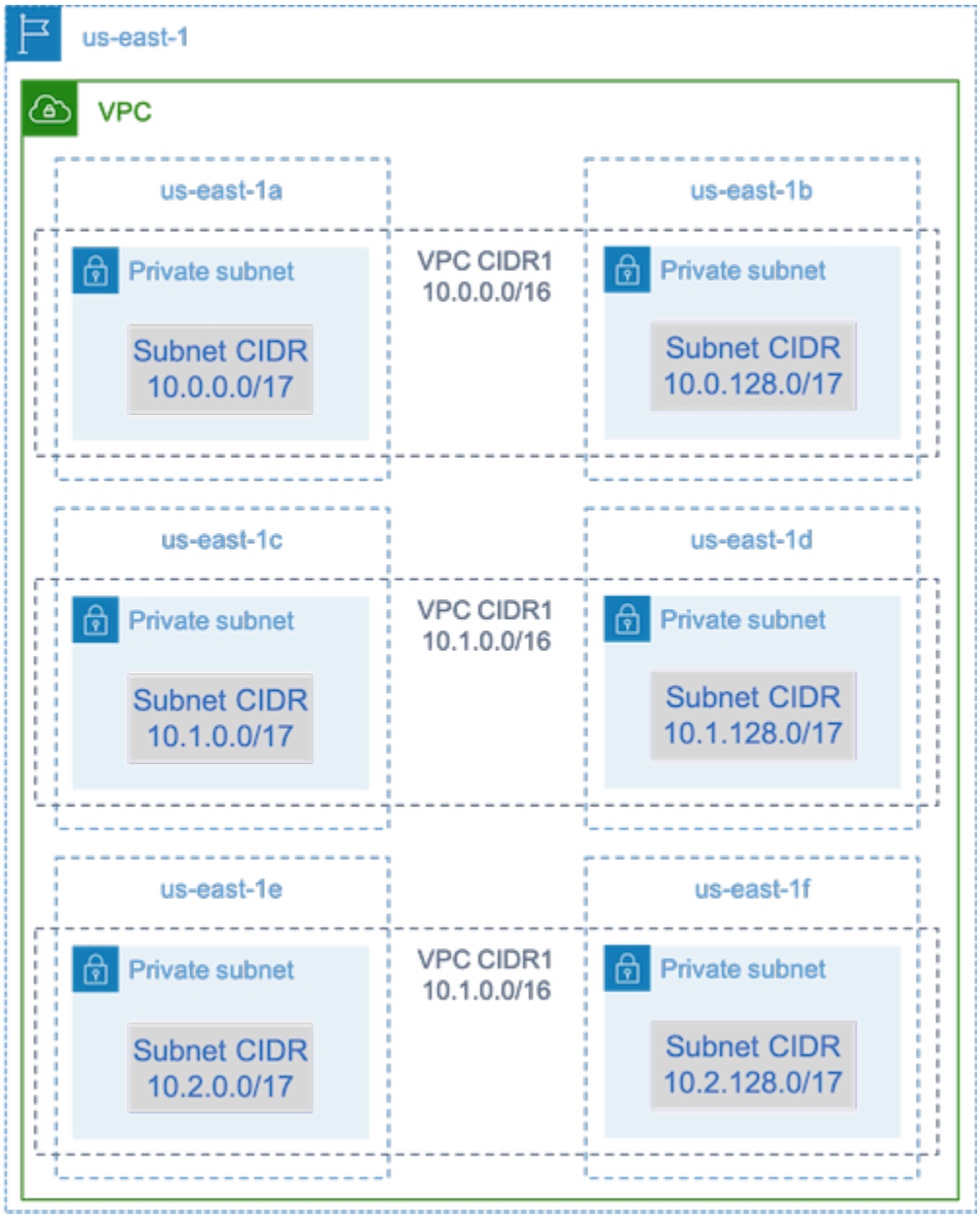
- 使用 Spot 中断控制面板 – 您可以使用 Spot 中断控制面板来跟踪 Spot 中断情况。应用程序提供有关已回收的 Amazon EC2 竞价型实例以及竞价型实例所在的可用区的指标。有关更多信息，请参阅 Spot 中断控制面板

常见错误和故障排除

AWS Batch 中的错误通常发生在应用程序级别，或者是由不符合您的特定作业要求的实例配置引起的。其他问题包括作业卡在 RUNNABLE 状态或计算环境陷入 INVALID 状态。有关故障排除在 RUNNABLE 状态中卡住的作业的更多信息，请参阅 作业在RUNNABLE状态卡住。有关对陷入 INVALID 状态的计算环境进行故障排除的信息，请参阅 INVALID 计算环境。

- 查看 Amazon EC2 Spot vCPU 限额 – 验证您当前的服务限额是否符合作业要求。例如，假设您当前的服务限额为 256 个 vCPU，而作业需要 10,000 个 vCPU。则服务限额不符合作业要求。有关更多信息和疑难解答说明，请参阅 Amazon EC2 服务限额和如何增加 Amazon EC2Resources 的服务限额？ 。
- 作业在应用程序运行之前失败 – 有些作业可能因为 DockerTimeoutError 错误或 CannotPullContainerError 错误而失败。有关疑难解答信息，请参阅如何解决 AWS Batch 中的"DockerTimeoutError"错误？。
- IP 地址不足 - 您的 VPC 和子网中的 IP 地址数量可能会限制您可以创建的实例数量。使用无类别域间路由，可提供多于运行工作负载所需的 IP 地址。如有必要，您还可以构建具有较大地址空间的专用 VPC。例如，您可以在 10.x.0.0/16 中创建一个包含多个 CIDR 的 VPC，并在每个可用区中创

建一个子网，CIDR 为 `10.x.y.0/17`。在此示例中，x 介于 1-4 之间，y 为 0 或 128。此配置在每个子网中提供 36,000 个 IP 地址。



- 验证实例是否已在 Amazon EC2 中注册 – 如果您在 Amazon EC2 控制台中看到您的实例，但在 Amazon ECS 集群中看不到 Amazon Elastic Container Service 容器实例，则可能未在亚马逊机器映像（AMI）上安装 Amazon ECS 代理。Amazon ECS 代理、AMI 中的 Amazon EC2 数据或启动模板也可能配置不正确。要找出根本原因，请创建单独的 Amazon EC2 实例或使用 SSH 连接到现有实例。有关更多信息，请参阅 Amazon ECS 容器代理配置、Amazon ECS 日志文件位置和 计算资源 &AMI;。
- 查看 AWS 控制面板 - 查看 AWS 控制面板以验证预期的作业状态以及计算环境是否按预期扩展。您也可以查看 CloudWatch 中的作业日志。

- 验证您的实例是否已创建 - 如果创建了实例，则意味着您的计算环境按预期进行扩展。如果您的实例未创建，请在计算环境中找到要更改的关联子网。有关更多信息，请参阅验证自动扩缩组的扩展活动。

  我们还建议您验证实例是否可以满足相关作业要求。例如，一项作业可能需要 1 TiB 的内存，但计算环境使用的 C5 实例类型限制为 192 GB 内存。
- 确认您的实例是由 AWS Batch 请求的 – 查看自动扩缩组历史记录以验证您的实例是否由 AWS Batch 请求过。这表明了 Amazon EC2 是如何尝试获取实例的。如果您收到错误消息，指出 Amazon EC2 Spot 无法在特定可用区获取实例，这可能是因为该可用区不提供特定的实例系列。
- 验证实例是否在 Amazon ECS 中注册 – 如果您在 Amazon EC2 控制台中看到实例，但在 Amazon ECS 集群中看不到任何 Amazon ECS 容器实例，则可能未在亚马逊机器映像（AMI）上安装 Amazon ECS 代理。此外，Amazon ECS 代理、AMI 中的 Amazon EC2 数据或启动模板可能配置不正确。要找出根本原因，请创建单独的 Amazon EC2 实例或使用 SSH 连接到现有实例。有关更多信息，请参阅 CloudWatch 代理配置文件：日志部分、Amazon ECS 日志文件位置 和 计算资源 &AMI;。
- 打开支持请求单 – 如果您在进行故障排除后仍遇到问题并且已经制定了支持计划，请打开支持请求单。在支持请求中，请务必包含有关问题、工作负载细节、配置和测试结果的信息。有关更多信息，请参阅比较 AWS Support 计划。
- 查看 AWS Batch 和 HPC 论坛 – 如需更多信息，请参阅AWS Batch和 HPC 论坛。
- 查看 AWS Batch 运行时监测控制面板 – 此控制面板使用无服务器架构捕获来自 Amazon ECS 的事件，AWS Batch，和 Amazon EC2 来提供对作业和实例的见解。有关更多信息，请参阅AWS Batch 运行时监控面板解决方案。

文档历史记录

下表描述了自首次发布以来对文档所做的重要更改 AWS Batch。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
更新了 AWS Batch 支持的亚马逊 EKS 版本	已更新 AWS Batch 支持移除版本 1.22 的 Amazon EKS 版本。	2024年3月11日
更新了 AWS Batch 支持的亚马逊 EKS 版本	已更新 AWS Batch 支持的 Amazon EKS 版本，使其包含版本 1.29。	2024 年 2 月 29 日
自动作业重试	更正了代码示例。	2024 年 2 月 29 日
增加了对多容器作业的支持 AWS Batch	增加了对亚马逊弹性容器服务、亚马逊 Elastic Kubernetes Service 和的多容器任务的支持。 AWS Batch AWS Fargate	2024年2月28日
更新了 AWS Batch 支持的亚马逊 EKS 版本	已将 AWS Batch 支持的 Amazon EKS 版本更新为包含版本 1.28	2024年1月27日
已更新BatchServiceRolePolicy 和 AWSBatchServiceRole	BatchServiceRolePolicy 已更新，增加了对描述 Spot 队列请求历史记录和 Amazon EC2 Auto Scaling 活动的支持。 AWSBatchServiceRole 已更新为添加语句 ID、向ec2:DescribeSpotFleetReques	2023 年 12 月 5 日

	tHistory 和授予 AWS Batch 权限autoscaling:DescribeScalingActivities 。	
AWS Batch 在亚马逊 EKS 上	AWS Batch 增加了对在 Amazon EKS 集群上运行作业的支持。	2022 年 10 月 25 日
跨服务混淆了副手的预防 AWS Batch	AWS Batch 现在为混乱的副手安全问题提供了一种解决方法，该问题是在一个实体（服务或帐户）被另一个实体强迫执行操作时出现的。	2022 年 6 月 6 日
接口 VPC 端点 (AWS PrivateLink)	增加了对配置由提供支持的接口 VPC 终端节点的支持 AWS PrivateLink。这意味着您可以在您的 VPC 之间创建私有连接，而 AWS Batch 无需通过 NAT 实例、VPN 连接或进行访问 AWS Direct Connect。	2022 年 4 月 15 日
增强的计算环境更新	AWS Batch 增强了对计算环境的支持更新。	2022 年 4 月 14 日
AWS 托管策略更新-更新现有策略	AWS Batch 更新了现有的托管策略。	2021 年 12 月 6 日
公平份额调度	AWS Batch 增加了对向任务队列添加调度策略的支持。	2021 年 11 月 9 日
Amazon EFS	AWS Batch 增加了对将 Amazon EFS 文件系统添加到您的任务定义的支持。	2021 年 4 月 1 日
添加了服务相关角色	AWS Batch 添加AWSServiceRoleForBatch服务相关角色。	2021 年 3 月 10 日

AWS Fargate 支持	AWS Batch 增加了对在 Fargate 资源上运行作业的支持。	2020 年 12 月 3 日
Amazon Linux 2 支持	AWS Batch 添加了对使用 EC2 配置参数在计算环境中自动选择 Amazon Linux 2 AMI 的支持。	2020 年 11 月 24 日
增强的重试策略	AWS Batch 增强了作业的重试策略。现在，作业可以重试，也可以通过模式匹配作业的`ExitCode`、`Reason`或`StatusReason` 来停止重试。	2020 年 10 月 20 日
为资源加标签	AWS Batch 增加了对向计算环境、作业定义、作业队列和作业添加元数据标签的支持。	2020 年 10 月 7 日
密文	AWS Batch 增加了对向作业传递机密的支持。	2020 年 10 月 1 日
日志记录	AWS Batch 添加了对为作业指定其他日志驱动程序的支持。	2020 年 10 月 1 日
分配策略	AWS Batch 添加了对多种策略的支持，以选择实例类型。	2019 年 10 月 16 日
EFA 支持	AWS Batch 增加了对弹性结构适配器 (EFA) Fabric Adapter 设备的支持。	2019 年 8 月 2 日
GPU 计划	AWS Batch 添加了 GPU 调度。使用此功能，可以指定每个作业所需的 GPU 数量，并相应地 AWS Batch 扩展实例。	2019 年 4 月 4 日

多节点并行作业	AWS Batch 增加了对多节点 parallel 作业的支持。可以使用此功能运行跨多个 Amazon EC2 实例的单个作业。	2018 年 11 月 19 日
资源级权限	AWS Batch 支持对多个 API 操作的资源级权限。	2018 年 11 月 12 日
Amazon EC2 启动模板支持	AWS Batch 增加了对在计算环境中使用启动模板的支持。	2018 年 11 月 12 日
AWS Batch 作业超时	AWS Batch 添加了对作业超时的支持。借助此支持，您可以为作业配置特定的超时时间，这样，如果作业运行的时间超过应有的时间，则 AWS Batch 终止该作业。	2018 年 4 月 5 日
AWS Batch 以就业为 EventBridge 目标	AWS Batch 就业机会被当作 EventBridge 目标。通过创建简单的规则，可以匹配事件并根据事件提交 AWS Batch 作业。	2018 年 3 月 1 日
CloudTrail 审计 AWS Batch	CloudTrail 可以审核对 AWS Batch API 操作的调用。	2018 年 1 月 10 日
数组作业	AWS Batch 添加了对阵列作业的支持。可以将数组作业用于参数扫描和蒙特卡罗工作负载。	2017 年 11 月 28 日
扩展了 AWS Batch 标记	AWS Batch 扩展了对标记功能的支持。可以使用此功能为在托管计算环境中启动的 Amazon EC2 竞价型实例指定标签。	2017 年 10 月 26 日

AWS Batch 的事件直播 EventBridge	AWS Batch 添加的事件流 EventBridge。您可以使用 AWS Batch 事件流接收有关提交到任务队列的任务状态的近乎实时的通知。	2017 年 10 月 24 日
自动作业重试	AWS Batch 添加了对任务重试的支持。通过此更新，可以在作业和作业定义中应用重试策略，以便在作业失败时自动重试。	2017 年 3 月 28 日
AWS Batch 正式上市	AWS Batch 已推出，旨在让您在上运行批量计算工作负载 AWS Cloud。	2017 年 1 月 5 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。