aws

用户指南

AWS Batch

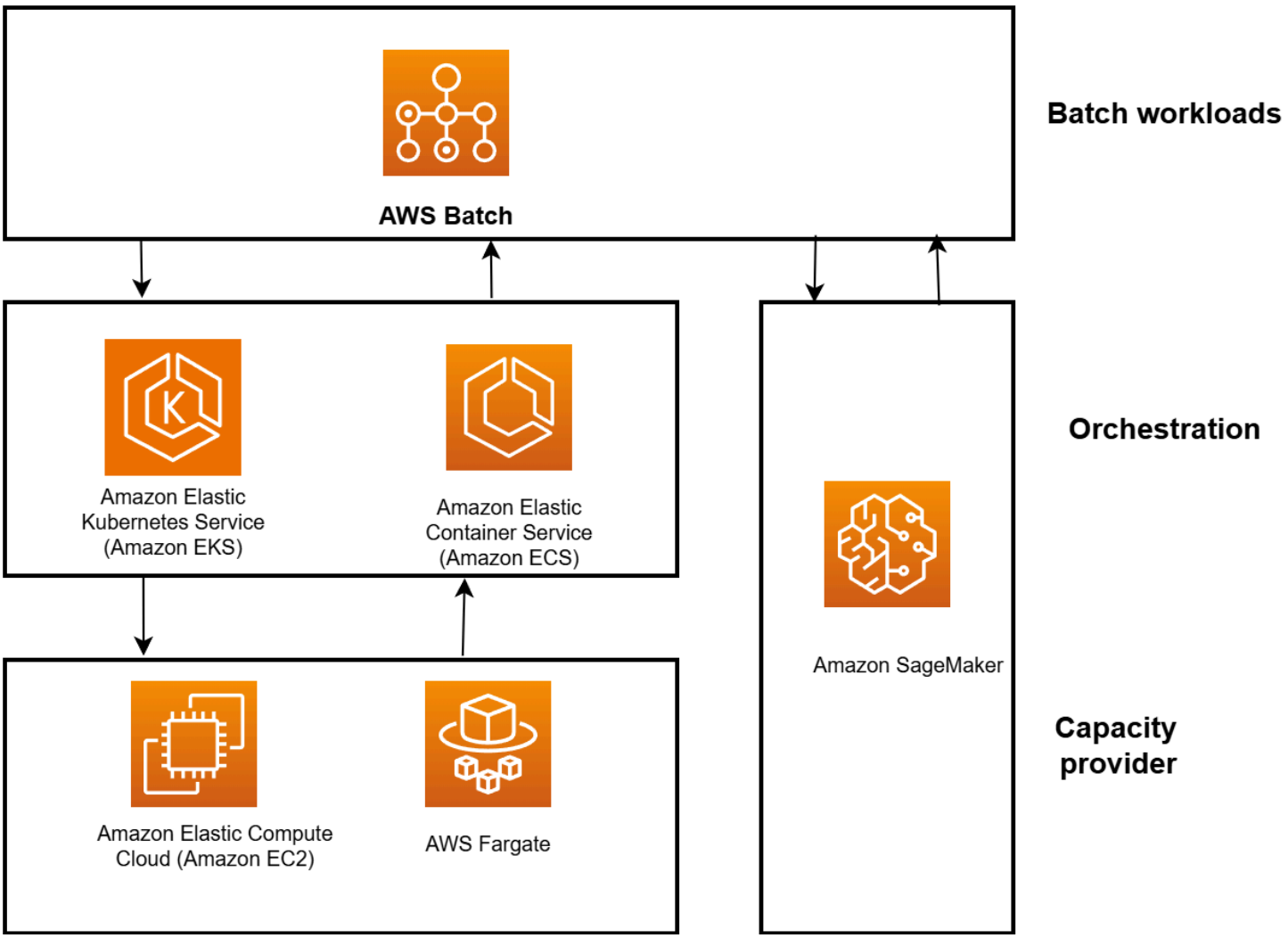
AWS Batch: 用户指南

Table of Contents

什么是 AWS Batch？

AWS Batch借助 ，您可以在 上运行批处理计算工作负载。AWS Cloud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。

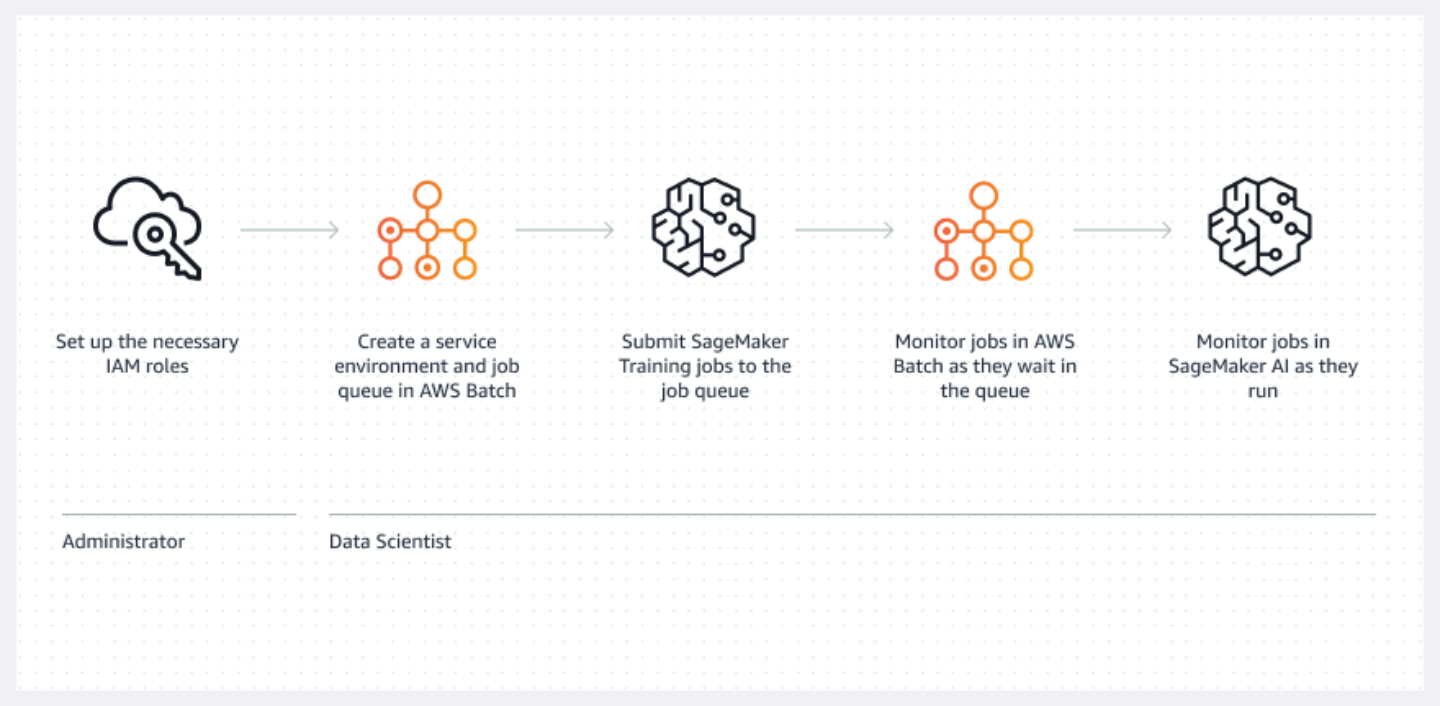


AWS Batch 在 AWS 托管式容器编排服务、Amazon ECS 和 Amazon EKS 的基础上提供了大规模运行计算密集型工作负载所需的所有必要功能。AWS Batch 能够扩展 Amazon EC2 实例和 Fargate 资源上的计算容量。

AWS Batch 为批处理工作负载提供了完全托管式的服务，此外还提供了多项操作功能来优化这些类型的工作负载，以满足吞吐量、速度、资源效率和成本的需要。

AWS Batch 还支持 SageMaker 训练作业排队功能，让数据科学家和机器学习工程师可以将具有优先级的训练作业提交到可配置的队列。此功能可确保机器学习工作负载在资源可用后立即自动运行，无需手动协调，同时还提高了资源利用率。

对于机器学习工作负载，AWS Batch 提供了适用于 SageMaker 训练作业的排队功能。您可以为队列配置特定的策略，来优化机器学习训练工作负载的成本、性能和资源分配。



这实现了管理员负责设置基础设施和权限，而数据科学家专注于提交和监控其机器学习训练工作负载的责任共担模式。作业会根据配置的优先级和资源可用性自动排队和执行。

您是 AWS Batch 新用户吗？

如果您是首次接触 AWS Batch 的用户，我们建议您先阅读以下部分：

- AWS Batch 的组成部分
- 创建 IAM 账户和管理员用户
- 设置 AWS Batch
- AWS Batch 教程入门
- SageMaker 人工智能 AWS Batch 入门

相关服务

AWS Batch 是一项完全托管式的批处理计算服务，可跨 AWS 计算服务（例如 Amazon ECS、Amazon EKS、AWS Fargate 以及竞价型或按需型实例）计划、调度和运行容器化批处理 ML、模拟和分析工作负载。有关各项托管式计算服务的更多信息，请参阅：

- Amazon EC2 用户指南
- AWS Fargate 开发人员指南
- Amazon EKS 用户指南
- Amazon SageMaker AI 开发人员指南

访问 AWS Batch

您可以通过以下方式访问 AWS Batch：

AWS Batch 管理控制台

用于创建和管理资源的 Web 界面。

AWS Command Line Interface

通过命令行 Shell 中的命令与 AWS 服务进行交互。AWS Command Line Interface 在 Windows、macOS 和 Linux 上受支持。有关 AWS CLI 的更多信息，请参阅 AWS Command Line Interface 用户指南。您可以在《AWS CLI Command Reference》https://docs.aws.amazon.com/cli/latest/reference/中查看 AWS Batch命令。

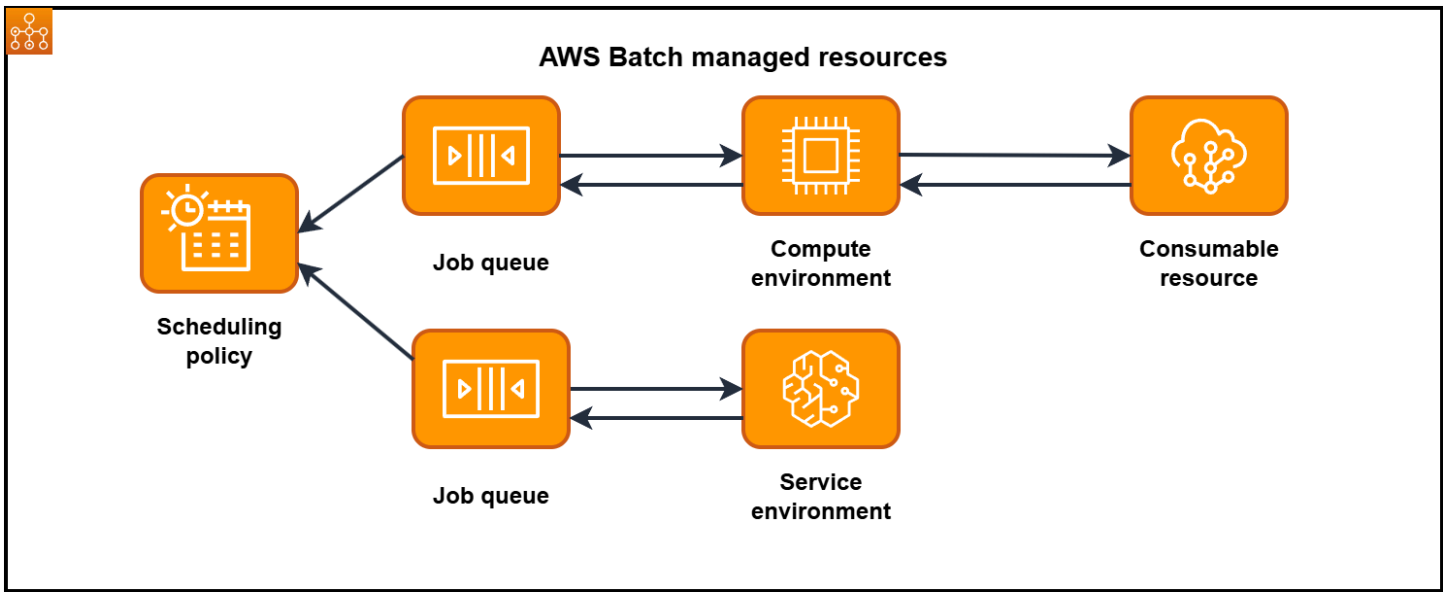
AWS SDK

如果您倾向于使用语言特定的 API 而非通过 HTTP 或 HTTPS 提交请求来构建应用程序，则可以使用 AWS 提供的库、示例代码、教程和其他资源。这些库提供了若干可自动执行任务的基本功能，例如以加密方式对请求进行签名、重试请求以及处理错误响应等。这些功能有助您更高效地入门。有关更多信息，请参阅用于在 AWS 上进行构建的工具。

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。



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 的计算环境 AWS Batch。

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。有关更多信息，请参阅 作业队列。

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

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)。作业可以按名称或按 ID 引用其他作业，并且可以依赖于其他作业的成功完成或您的指定的资源的可用性。有关更多信息，请参阅 作业。

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。使用公平份额调度策略，您可以为工作负载或用户分配不同的份额标识符。AWS Batch 作业调度器默认使用先进先出（FIFO）策略。有关更多信息，请参阅 公平份额调度策略。

消耗性资源

消耗性资源是运行作业所需的资源，例如第三方许可证令牌、数据库访问带宽、对第三方 API 的调用进行节流的需求等。您可以指定作业运行所需的消耗性资源，Batch 在调度作业时会考虑这些资源依赖项。您可以通过仅分配具有全部所需资源的作业来减少计算资源利用不足的情况。有关更多信息，请参阅 资源感知调度。

服务环境

服务环境定义了 AWS Batch 会如何与 SageMaker 集成以执行作业。服务环境让 AWS Batch 能够在 SageMaker 上提交和管理作业，同时还提供 AWS Batch 的排队、调度和优先级管理功能。服务环境定义了特定服务类型（例如 SageMaker 训练作业）的容量限制。容量限制用于控制环境中服务作业可使用的最大资源量。有关更多信息，请参阅 的服务环境 AWS Batch。

服务作业

服务作业是您提交到 AWS Batch 以在服务环境中运行的一个工作单位。服务作业利用 AWS Batch 的排队和调度功能，同时将实际执行委托给外部服务。例如，作为服务作业提交的 SageMaker 训练作业由 AWS Batch 排队和管理优先级，但 SageMaker 训练作业将在 SageMaker AI 基础设施中执行。这种集成使数据科学家和机器学习工程师能够将用 AWS Batch 的自动工作负载管理和优先级排队等功能用于其 SageMaker AI 训练工作负载。服务作业可以按名称或 ID 引用其他作业，并支持作业依赖项。有关更多信息，请参阅 中的服务职位 AWS Batch。

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 AWS Batch 在其计算环境中使用了 Amazon ECS 容器实例。要在 AWS CLI 中使用 AWS Batch，必须使用支持最新 AWS CLI 功能的 AWS Batch 版本。如果在 AWS CLI 中没有看到对 AWS Batch 功能的支持，可以升级到最新版本。有关更多信息，请参阅 http://aws.amazon.com/cli/。

> **Note**
>
> 由于 AWS Batch 使用了 Amazon EC2 的组件，您可以将 Amazon EC2 控制台用于这些步骤中的许多步骤。

要开始设置 AWS Batch，请完成以下任务。

主题

- 创建 IAM 账户和管理员用户
- 为您的计算环境和容器实例创建 IAM 角色
- 为实例创建密钥对
- 创建 VPC
- 创建安全组
- 安装 AWS CLI

创建 IAM 账户和管理员用户

首先，您需要创建一个 AWS 帐户和一个通常被授予管理权限的用户。要完成此操作，请完成以下教程：

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。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，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。作为最佳安全实践，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。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您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 管理控制台在下一页上，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 Signing in as the root user。
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 Enabling。 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 URL。

  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 Create a permission set。
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 Add groups。

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 凭证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 AWS Identity and Access Management 角色，然后将该角色与计算环境关联。

> **Note**
>
> 要验证您的 AWS 账户 是否拥有所需的权限，请参阅为您的账户设置的初始 IAM 服务。在控制台首次运行体验中将自动为您创建 AWS Batch 计算环境和容器实例角色。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。如果您计划改用 AWS CLI，请先完成将服务关联角色用于 AWS Batch、Amazon ECS 实例角色和教程：创建 IAM 执行角色中的过程，然后再创建您的第一个计算环境。

为实例创建密钥对

AWS 使用公有密钥密码术来保护实例的登录信息。Linux 实例（例如 AWS Batch 计算环境容器实例）没有用于 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，如果您计划在多个 AWS 区域 中启动实例，请在每个区域中创建密钥对。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. 从导航栏中，选择密钥对的 AWS 区域。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
   ```

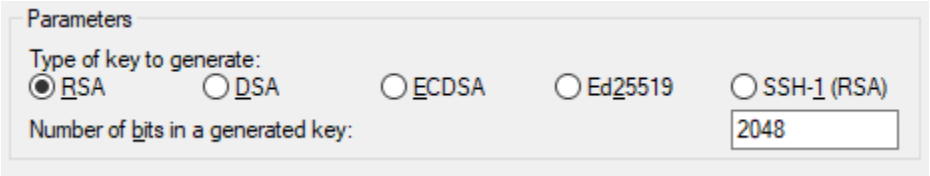
有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 密钥对。

使用密钥对连接到实例

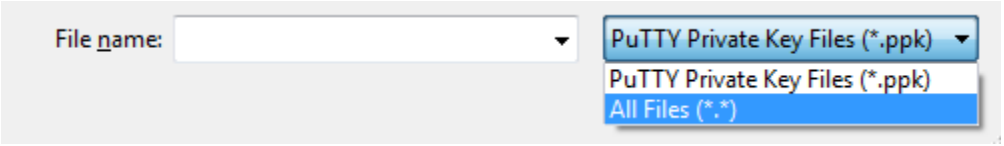
要从运行 Mac 或 Linux 的计算机连接到 Linux 实例，需要使用 .pem 选项对 SSH 客户端指定 -i 文件和私有密钥的路径。要从运行 Windows 的计算机连接到您的 Linux 实例，请使用 MindTerm 或 PuTTY。如果您计划使用 PuTTY，请安装它并遵循以下过程将 .pem 文件转换为 .ppk 文件。

（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

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。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。要确定您是否具有默认 VPC，请参阅《Amazon 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
租赁	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？。

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，请参阅 >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

要对 AWS CLI 使用 AWS Batch，请安装最新 AWS CLI 版本。有关安装 AWS CLI 或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

AWS Batch 教程入门

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。

之后，您可以使用 AWS Batch 首次运行向导创建计算环境、作业队列并提交 Hello World 作业示例。

通过向导开始使用 Amazon EC2 编排功能

Amazon Elastic Compute Cloud(Amazon EC2)在 中提供可扩展的计算容量AWS Cloud 使用 Amazon EC2 可避免前期的硬件投入，使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。

概览

本教程演示如何使用向导设置 AWS Batch，以配置 Amazon EC2 并运行 `Hello World`。

目标受众

本教程面向负责设置、测试和部署 AWS Batch 的系统管理员和开发人员而设计。

使用的功能

本教程介绍如何通过 AWS Batch 控制台向导来完成以下操作：

- 创建和配置 Amazon EC2 计算环境
- 创建作业队列。
- 创建任务定义
- 创建并提交要运行的作业
- 在 CloudWatch 中查看作业的输出

所需时间

完成本教程大约需要 10–15 分钟。

区域限制

使用此解决方案没有任何国家或地区限制。

资源用量费用

创建 AWS 账户并不会收费；但是，通过实施此解决方案，您可能会产生下表中列出的部分或全部费用。

描述	费用（美元）
Amazon EC2 实例	您需要为创建的每个 Amazon EC2 实例付费。有关定价的更多信息，请参阅 Amazon EC2 定价。

先决条件

开始前的准备工作：

- 如果您还没有 AWS 账户，请先创建一个。
- 创建 `ecsInstanceRole`实例角色。

第 1 步：创建计算环境

> **Important**
>
> 本教程包含的步骤均使用默认设置，以助您尽可能简单快速地入门。在出于生产用途而开始创建之前，我们建议您熟悉所有设置，并使用符合您要求的设置进行部署。

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. 对于创建作业和编排类型，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，选择一个附加了所需 IAM 权限的现有实例角色。该实例角色会允许计算环境中的 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。

该实例角色的默认名称为 ecsInstanceRole。

6. 对于实例配置，您可以保留默认设置。
7. 对于网络配置，使用您在该 AWS 区域的默认 VPC。
8. 选择下一步。

第 2 步：创建作业队列

作业队列存储您提交的作业，直到 AWS Batch 调度器在您的计算环境中的资源上运行作业为止。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. 对于作业队列配置中的名称，请为您的作业队列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. 对于所有其他配置选项，您可以保留默认值。
3. 选择下一步。

步骤 3：创建作业定义

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。

创建作业定义：

1. 对于创建作业定义

   a. 对于名称，请为您的作业队列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. 对于命令 - 可选，您可以将 hello world 更改为某个自定义消息，也可保留原样。
2. 对于所有其他配置选项，您可以保留默认值。
3. 选择下一步。

第 4 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. 对于所有其他配置选项，您可以保留默认值。
3. 选择下一步。

第 5 步：审核并创建

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。

1. 在审核和创建中，选择创建资源。
2. 在 AWS Batch 开始分配资源时会打开一个窗口。完成后，选择转到控制面板。您应会在该控制面板上看到所有已分配的资源并且该作业处于 `Runnable` 状态。您的作业将按计划运行，应会在 2-3 分钟内完成。

第 6 步：查看作业输出

要查看作业输出，请执行以下操作：

1. 在导航窗格中，选择作业。
2. 在作业队列下拉列表中，选择您为本教程创建的作业队列。
3. 作业表会列出您的所有作业及其当前状态。作业的状态变为成功后，选择该作业的名称以查看该作业的详细信息。
4. 在详细信息窗格中，选择日志流名称。这时将会打开该作业的 CloudWatch 控制台，并且应会有一个消息为 `hello world` 或自定义消息的事件。

第 7 步：清理教程资源

启用 Amazon EC2 实例后，您需要为该实例付费。您可以删除实例以停止产生费用。

要删除您创建的资源，请执行以下操作：

1. 在导航窗格中，选择作业队列。
2. 在作业队列表中，选择您为本教程创建的作业队列。
3. 选择 禁用。当作业队列的状态变为“已禁用”后，您可以选择删除。
4. 删除该作业队列后，在导航窗格中选择计算环境。

5. 选择您为本教程创建的计算环境，然后选择禁用。计算环境的禁用可能需要 1-2 分钟才能完成。
6. 计算环境的状态变为“已禁用”后，选择删除。计算环境的删除可能需要 1-2 分钟才能完成。

其他资源

完成本教程后，您可以探索以下主题：

- 探索 AWS Batch 的核心组件。有关更多信息，请参阅 AWS Batch 的组成部分。
- 详细了解 AWS Batch 支持的各种计算环境。
- 详细了解作业队列及其各种调度选项。
- 详细了解作业定义以及各种配置选项。
- 详细了解各种不同的作业类型。

开始使用向导 AWS Batch 和 Fargate 编排

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。

概述

本教程演示如何使用向导 AWS Batch 进行设置以配置 AWS Fargate 并运行。`Hello World`

目标受众

本教程面向负责设置、测试和部署 AWS Batch的系统管理员和开发人员而设计。

使用的功能

本教程向您展示如何使用 AWS Batch 控制台向导执行以下操作：

- 创建和配置 AWS Fargate 计算环境
- 创建作业队列。
- 创建作业定义
- 创建并提交要运行的作业
- 在中查看作业的输出 CloudWatch

所需时间

完成本教程大约需要 10-15 分钟。

区域限制

使用此解决方案没有任何国家/地区或区域限制。

资源使用成本

创建 AWS 账户不收取任何费用。但是，通过实施此解决方案，您可能会产生下表中列出的部分或全部费用。

说明	费用（美元）
定价基于任务或容器组（pod）请求的 vCPU、内存、操作系统、CPU 架构和存储资源。	有关定价的更多信息，请参阅 Fargate 定价。

先决条件

开始前的准备工作：

- AWS 账户 如果没有，请创建一个。
- 创建任务执行角色：如果您尚未创建任务执行角色，则可以在本教程中创建该角色。

第 1 步：创建计算环境

> **Important**
>
> 本教程包含的步骤均使用默认设置，以助您尽可能简单快速地入门。在出于生产用途而开始创建之前，我们建议您熟悉所有设置，并使用符合您要求的设置进行部署。

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. 对于配置作业和编排类型，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. 对于所有其他配置选项，您可以保留默认值。
6. 选择下一步。

第 2 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 900。
3. 对于所有其他配置选项，您可以保留默认值。
4. 选择下一步。

步骤 3：创建作业定义

创建作业定义：

1. 在常规配置部分：
   - 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（_）。
2. 在 Fargate 平台配置部分中：
   a. 开启分配公有 IP 以分配一个公有 IP 地址。除非您已设置私有映像存储库，否则您需要有公有 IP 才能下载容器映像。
   b. 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。选择ecsTaskExecution角色或BatchEcsTaskExecutionRole。

      要创建该执行角色，请选择创建执行角色。在创建 IAM 角色模态中，选择创建 IAM 角色。

      i. IAM 控制台已配置了用于创建该执行角色的权限设置。
      ii. 对于可信实体类型，请确认已选择了 AWS 服务。
      iii. 对于服务或用户案例，请确认已选择了 Elastic Container Service。
      iv. 选择下一步。

v. 对于权限策略，请确认已选择 ECSTaskExecutionRolePolicyAmazon。

vi. 选择下一步。

vii. 对于 “名称”，请查看并创建，验证角色名称是否为BatchEcsTaskExecutionRole。

viii. 选择创建角色。

ix. 在 AWS Batch 控制台中，选择执行角色旁边的刷新按钮。选择BatchEcsTaskExecutionRole执行角色。

3. 在容器配置部分：

   - 对于命令，您可以将 `hello world` 更改为某个自定义消息，也可保留原样。

4. 对于所有其他配置选项，您可以保留默认值。

5. 选择下一步。

第 4 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. 对于所有其他配置选项，您可以保留默认值。

3. 选择下一步。

第 5 步：审核并创建

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。

第 6 步：查看作业输出

要查看作业输出，请执行以下操作：

1. 在导航窗格中，选择作业。

2. 在作业队列下拉列表中，选择您为本教程创建的作业队列。

3. 作业表会列出您的所有作业及其当前状态。作业的状态变为成功后，选择该作业的名称以查看该作业的详细信息。

4. 在详细信息窗格中，选择日志流名称。Job 的 CloudWatch 控制台将打开，并且应该有一个带有消息`hello world`或您的自定义消息的事件。

第 7 步：清理教程资源

启用 Amazon EC2 实例后，您需要为该实例付费。您可以删除实例以停止产生费用。

要删除您创建的资源，请执行以下操作：

1. 在导航窗格中，选择作业队列。
2. 在作业队列表中，选择您为本教程创建的作业队列。
3. 选择禁用。当作业队列的状态变为“已禁用”后，您可以选择删除。
4. 删除该作业队列后，在导航窗格中选择计算环境。
5. 选择您为本教程创建的计算环境，然后选择禁用。计算环境的禁用可能需要 1-2 分钟才能完成。
6. 计算环境的状态变为“已禁用”后，选择删除。计算环境的删除可能需要 1-2 分钟才能完成。

其他资源

完成本教程后，您可以探索以下主题：

- 详细了解最佳实践。
- 探索 AWS Batch 核心组件。有关更多信息，请参阅 AWS Batch 的组成部分。
- 详细了解 AWS Batch支持的各种计算环境。
- 详细了解作业队列及其各种调度选项。
- 详细了解作业定义以及各种配置选项。
- 详细了解各种不同的作业类型。

入门 AWS Batch 和 Fargate 正在使用 AWS CLI

本教程演示如何设置 AWS Batch AWS Fargate 编排并使用 AWS Command Line Interface (AWS CLI) 运行简单的 “Hello World” 作业。您将了解如何创建计算环境、作业任务队列、作业定义以及如何将作业提交到 AWS Batch。

主题

- 先决条件

先决条件

在开始本教程之前，请确保您具有以下各项。

1. 的 AWS CLI。如需安装，请遵循 AWS CLI 安装指南。您也可以使用 AWS CloudShell，其中包括 AWS CLI。
2. 为你配置 AWS CLI 了相应的凭证。如果尚未设置凭证，请运行 `aws configure`。
3. 基本熟悉命令行界面和容器化概念。
4. 如何 AWS Batch 与 IAM 配合使用在中创建和管理 AWS Batch 资源、IAM 角色和 VPC 资源 AWS 账户。
5. 来自您的 VPC 的子网 ID 和安全组 ID AWS 账户。如果您还没有默认 VPC，则可以创建一个。有关使用检索这些资源的 IDs更多信息，请参阅 AWS CLI desc ribe-subnet s 和《命令参考describe-security-groups》。AWS CLI

所需时间：完成本教程大约需要 15-20 分钟时间。

成本：本教程将使用 Fargate 计算资源。假设您在完成本教程后立即按照清理说明删除相关资源，则完成本教程的成本估计不到 0.01 美元。Fargate 根据使用的 vCPU 和内存资源按秒收费，不足 1 分钟按 1 分钟计算。有关现行定价信息，请参阅 AWS Fargate 定价。

创建 IAM 执行角色

AWS Batch 需要一个允许亚马逊弹性容器服务 (Amazon ECS) 代理代表 AWS 您进行 API 调用的执行角色。Fargate 任务需要此角色才能提取容器镜像并将日志写入亚马逊。 CloudWatch

创建信任策略文档

首先，创建一个允许 Amazon EKS 任务服务代入该角色的信任策略。

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

创建执行角色

以下命令会使用您刚才创建的信任策略创建一个名为 BatchEcsTaskExecutionRoleTutorial 的 IAM 角色。

```
aws iam create-role \
    --role-name BatchEcsTaskExecutionRoleTutorial \
    --assume-role-policy-document file://batch-execution-role-trust-policy.json
```

附加所需的策略

附上为执行 Amazon ECS 任务提供必要权限的 AWS 托管策略。

```
aws iam attach-role-policy \
    --role-name BatchEcsTaskExecutionRoleTutorial \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
```

该角色现已准备就绪，可以用来执行 AWS Batch Fargate 任务。

创建计算环境

计算环境定义了批处理作业将在其中运行的计算资源。在本教程中，您将创建一个可根据作业要求自动预调配和扩展资源的托管式 Fargate 计算环境。

创建计算环境

以下命令会创建一个 Fargate 计算环境。将示例子网和安全组 IDs 替换为您自己的子网和安全组先决条件。

```
aws batch create-compute-environment \
    --compute-environment-name my-fargate-compute-env \
    --type MANAGED \
    --state ENABLED \
    --compute-resources type=FARGATE,maxvCpus=128,subnets=subnet-
a123456b,securityGroupIds=sg-a12b3456
```

以下展示了该命令成功运行时的输出。

```
{
    "computeEnvironmentName": "my-fargate-compute-env",
    "computeEnvironmentArn": "arn:aws:batch:us-west-2:123456789012:compute-environment/
my-fargate-compute-env"
}
```

等待计算环境准备就绪

检查计算环境的状态以确保其已准备就绪，然后再继续操作。

```
aws batch describe-compute-environments \
    --compute-environments my-fargate-compute-env \
    --query 'computeEnvironments[0].status'
```

```
"VALID"
```

当状态显示为 VALID 时，即表示计算环境已准备就绪，可以接受作业。

创建作业队列

作业队列存储已提交的作业，直到 AWS Batch 调度器在您的计算环境中的资源上运行这些作业。队列中的作业按优先级顺序处理。

创建作业队列

以下命令会创建一个优先级为 900 并使用您的 Fargate 计算环境的作业队列。

```
aws batch create-job-queue \
    --job-queue-name my-fargate-job-queue \
    --state ENABLED \
    --priority 900 \
    --compute-environment-order order=1,computeEnvironment=my-fargate-compute-env
```

以下展示了该命令成功运行时的输出。

```
{
    "jobQueueName": "my-fargate-job-queue",
    "jobQueueArn": "arn:aws:batch:us-west-2:123456789012:job-queue/my-fargate-job-
queue"
}
```

验证作业队列是否准备就绪

检查您的作业队列是否已处于 ENABLED 状态并准备就绪，可以接受作业。

```
aws batch describe-job-queues \
    --job-queues my-fargate-job-queue \
    --query 'jobQueues[0].state' "ENABLED"
```

创建作业定义

作业定义用于指定作业的运行方式，包括要使用的 Docker 映像、资源要求和其他参数。对于 Fargate，您将使用资源要求而不是传统的 vCPU 和内存参数。

创建作业定义

以下命令会创建一个将使用 busybox 容器映像来运行简单的“hello world”命令的作业定义。123456789012用你的实际 AWS 账户 身份证替换，用你自己的身份证 AWS 区域 替换示例。

```
aws batch register-job-definition \
```

```
    --job-definition-name my-fargate-job-def \
    --type container \
    --platform-capabilities FARGATE \
    --container-properties '{
        "image": "busybox",
        "resourceRequirements": [
            {"type": "VCPU", "value": "0.25"},
            {"type": "MEMORY", "value": "512"}
        ],
        "command": ["echo", "hello world"],
        "networkConfiguration": {
            "assignPublicIp": "ENABLED"
        },
        "executionRoleArn": "arn:aws:iam::123456789012:role/
BatchEcsTaskExecutionRoleTutorial"
    },
{
    "jobDefinitionName": "my-fargate-job-def",
    "jobDefinitionArn": "arn:aws:batch:us-west-2:123456789012:job-definition/my-
fargate-job-def:1",
    "revision": 1
}'
```

该作业定义指定了 0.25 个 vCPU 和 512 MB 内存，这是 Fargate 任务所需的最低资源。assignPublicIp 设置已启用，因此该容器可以从 Docker Hub 中提取 busybox 映像。

提交并监控作业

现在您已拥有所有必需的组件，可以向队列提交作业并监控其进度。

提交一份工作

以下命令会使用您创建的作业定义将一个作业提交到您的队列。

```
aws batch submit-job \
    --job-name my-hello-world-job \
    --job-queue my-fargate-job-queue \
    --job-definition my-fargate-job-def
```

以下展示了该命令成功运行时的输出。

```
{
    "jobArn": "arn:aws:batch:us-west-2:123456789012:job/my-hello-world-job",
```

```
    "jobName": "my-hello-world-job",
    "jobId": "1509xmpl-4224-4da6-9ba9-1d1acc96431a"
}
```

记下响应中返回的 jobId，您将使用它来监控作业的进度。

监控作业状态

使用作业 ID 来检查作业的状态。作业将经过多个状
态：SUBMITTED、PENDING、RUNNABLE、STARTING、RUNNING 以及最后的 SUCCEEDED 或 FAILED。

```
aws batch describe-jobs --jobs 1509xmpl-4224-4da6-9ba9-1d1acc96431a
```

以下展示了该命令成功运行时的输出。

```
{
    "jobs": [
        {
            "jobArn": "arn:aws:batch:us-west-2:123456789012:job/my-hello-world-job",
            "jobName": "my-hello-world-job",
            "jobId": "1509xmpl-4224-4da6-9ba9-1d1acc96431a",
            "jobQueue": "arn:aws:batch:us-west-2:123456789012:job-queue/my-fargate-job-
queue",
            "status": "SUCCEEDED",
            "createdAt": 1705161908000,
            "jobDefinition": "arn:aws:batch:us-west-2:123456789012:job-definition/my-
fargate-job-def:1"
        }
    ]
}
```

当状态显示为 SUCCEEDED 时，即表示您的作业已成功完成。

查看作业输出

任务完成后，您可以在 Amazon Logs 中查看其输 CloudWatch 出。

获取日志流名称

首先，从作业详细信息中检索日志流名称。请将示例作业 ID 替换为您自己的值。

```
aws batch describe-jobs --jobs 1509xmpl-4224-4da6-9ba9-1d1acc96431a \
```

```
    --query 'jobs[0].attempts[0].containers[0].logStreamName' \
    --output text
```

```
my-fargate-job-def/default/1509xmpl-4224-4da6-9ba9-1d1acc96431a
```

查看作业日志

使用日志流名称从 CloudWatch Logs 中检索作业的输出。

```
aws logs get-log-events \
    --log-group-name /aws/batch/job \
    --log-stream-name my-fargate-job-def/default/1509xmpl-4224-4da6-9ba9-1d1acc96431a \
    --query 'events[*].message' \
    --output text
```

输出会显示“hello world”，确认您的作业已成功运行。

清理 资源

为避免持续产生费用，请清理您在此教程中创建的资源：由于依赖项的原因，您必须按正确的顺序删除资源。

禁用和删除作业队列

首先，禁用作业队列，然后将其删除。

```
aws batch update-job-queue \
    --job-queue my-fargate-job-queue \
    --state DISABLED
```

```
aws batch delete-job-queue \
    --job-queue my-fargate-job-queue
```

禁用和删除计算环境

删除作业队列后，请禁用并删除计算环境。

```
aws batch update-compute-environment \
    --compute-environment my-fargate-compute-env \
    --state DISABLED
```

```
aws batch delete-compute-environment \
    --compute-environment my-fargate-compute-env
```

清理 IAM 角色

移除附加的策略并删除该 IAM 角色。

```
aws iam detach-role-policy \
    --role-name BatchEcsTaskExecutionRoleTutorial \
    --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy
```

```
aws iam delete-role \
    --role-name BatchEcsTaskExecutionRoleTutorial
```

移除临时文件

删除您创建的信任策略文件。

```
rm batch-execution-role-trust-policy.json
```

所有资源都已成功清理。

投入生产

本教程旨在帮助您了解 Fargate AWS Batch 的工作原理。对于生产部署，应注意以下附加要求：

安全性注意事项：

- 创建具有所需最低访问权限的专用安全组，而不要使用默认安全组
- 使用带有 NAT 网关的私有子网，而不是为容器分配的公有 IP
- 将容器映像存储在 Amazon ECR 中，而不要使用公有存储库
- 为 AWS 服务通信实现 VPC 终端节点，以避开互联网流量

架构注意事项：

- 跨多个可用区部署以确保高可用性
- 实施作业重试策略和死信队列，以用于错误处理
- 使用具有不同优先级的多个作业队列，以用于工作负载管理

- 配置基于根队列深度和资源利用率的自动扩缩策略
- 实施作业失败和资源利用率监控和提醒

运营注意事项：

- 设置用于监控的 CloudWatch 仪表板和警报
- 实施恰当的日志记录和审计跟踪记录
- 将 o CloudFormation r AWS CDK 用于基础架构即代码
- 建立备份和灾难恢复程序

有关生产就绪架构的全面指导，请参阅 AWS Well-Architected 框架和 AWS Security Best Practices。

后续步骤

现在您已经完成了本教程，可以探索更多高级 AWS Batch 功能：

- 作业队列：了解作业队列调度和优先级管理
- 作业定义：探索高级作业定义配置，包括环境变量、卷和重试策略
- 的计算环境 AWS Batch：了解各种不同的计算环境类型和扩缩选项
- 多节点并行作业：运行跨多个计算节点的作业
- 数组作业：高效提交大量相似的作业
- AWS Batch 的最佳做法：了解适用于生产工作负载的优化技术

开始使用 Amazon EKS 上的 AWS Batch

Amazon EKS 上的 AWS Batch 是一项托管服务，用于将批处理工作负载调度和扩展到现有 Amazon EKS 集群中。AWS Batch 不会代表您创建、管理或执行您的 Amazon EKS 集群生命周期操作。AWS Batch 编排纵向扩展和缩减由 AWS Batch 管理的节点，并在这些节点上运行容器组（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、自动扩缩节点组或容器组（pod）生命周期。为了 AWS Batch 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，它可以在其中将容器组（pod）限定为 AWS Batch 作业。我们建议使用专用的命名空间将 AWS Batch 容器组（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 CreateComputeEnvironment API 操作将该 Amazon EKS 集群关联到 AWS Batch 计算环境。作业队列可以与这个新的 Amazon EKS 计算环境相关联。使用 SubmitJob API 操作基于 Amazon EKS 作业定义将 AWS Batch 作业提交到作业队列。然后 AWS Batch 启动 AWS Batch 托管节点，并将作业队列中的作业作为 Kubernetes 容器组（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 上设置好 AWS Batch。

目录

概览

本教程展示了如何使用 AWS CLI、`kubectl` 和 `eksctl` 设置将 AWS Batch 与 Amazon EKS 结合使用。

目标受众

本教程面向负责设置、测试和部署 AWS Batch 的系统管理员和开发人员而设计。

使用的功能

本教程介绍如何使用 AWS CLI 来执行以下操作：

- 创建和配置 Amazon EKS 计算环境
- 创建作业队列。

- 创建任务定义
- 创建并提交要运行的作业
- 提交带覆盖的作业

所需时间

完成本教程大约需要 30–40 分钟。

区域限制

使用此解决方案没有任何国家或地区限制。

资源用量费用

创建 AWS 账户并不会收费；但是，通过实施此解决方案，您可能会产生下表中列出的部分或全部费用。

描述	费用（美元）
您需要按集群小时数付费	因实例而异，详情请参阅 Amazon EKS 定价

先决条件

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

- 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。在安装 AWS CLI 后，建议您还要对其进行配置。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 `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`** – 用于处理 Amazon EKS 集群的命令行工具，该工具可自动执行许多单独任务。本指南要求您使用 `0.115.0` 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`。
- 所需的 IAM 权限 – 您正在使用的 IAM 安全主体必须具有使用 Amazon EKS IAM 角色和服务相关角色的权限、CloudFormation 以及 VPC 和相关资源。有关更多信息，请参阅《IAM 用户指南》中

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

- 权限 – 调用 CreateComputeEnvironment API 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。
- AWS 账户账号：您需要知道您的 AWS 账户 ID。按照查看您的 AWS 账户 ID 中的说明进行操作。
- （可选）CloudWatch ：要检查（可选）提交带覆盖的作业的详细信息，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。

第 1 步：为 AWS Batch 创建 Amazon EKS 集群

> **Important**
>
> 本教程包含的步骤均使用默认设置，以助您尽可能简单快速地入门。在出于生产用途而开始创建之前，我们建议您熟悉所有设置，并使用符合您要求的设置进行部署。

完成前提条件安装后，您需要使用 `eksctl` 创建集群。集群创建可能需要 10–15 分钟时间。

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

请在上述命令中执行以下替换：

- 将 *my-cluster-name* 替换为要用于集群的名称。
- 将 *region-code* 替换为要在其中创建集群的 AWS 区域，例如 `us-west-2`。

本教程的后面部分将需要该集群名称和区域。

第 2 步：为 AWS Batch 准备您的 Amazon EKS 集群

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 EKS 集群执行一次此操作。

```
$ 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: ["events"]
    verbs: ["list"]
  - 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
```

3. 为 AWS Batch 创建名称空间范围的 Kubernetes 角色，用于管理和绑定生命周期容器组（pod）。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
```

4. 更新 Kubernetes `aws-auth` 配置映射以将前面的 RBAC 权限映射到 AWS Batch 服务相关角色。

   请在以下命令中执行以下替换：

   - 将 *<your-account-number>* 替换为您自己的 AWS 账户账号。

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

   输出：

```
2022-10-25 20:19:57 [#]  adding identity "arn:aws:iam::<your-account-number>: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 aws-authconfigmap 中带路径的角色在其 ARN 中包含路径时不起作用。

第 3 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 用以管理您的 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。

由于 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。首先，创建一个指向 Amazon EKS 集群的计算环境。

- 对于 subnets，请运行 `eksctl get cluster` *my-cluster-name* 来获取集群使用的子网。
- 对于 securityGroupIds 参数，您可以使用与 Amazon EKS 集群相同的安全组。此命令检索集群的安全组 ID。

```
$ aws eks describe-cluster \
    --name my-cluster-name \
    --query cluster.resourcesVpcConfig.clusterSecurityGroupId
```

- instanceRole 是在您创建集群时创建的。要查找 instanceRole，请列出使用 AmazonEKSWorkerNodePolicy 策略的所有实体：

```
$  aws iam list-entities-for-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy
```

  该策略角色的名称包含您创建的集群的名称 eksctl-*my-cluster-name*-nodegroup-example。

  要查找 instanceRole ARN，请运行以下命令：

```
$  aws iam list-instance-profiles-for-role --role-name eksctl-my-cluster-name-
nodegroup-example
```

输出：

```
INSTANCEPROFILES        arn:aws:iam::<your-account-number>:instance-profile/
eks-04cb2200-94b9-c297-8dbe-87f12example
```

有关更多信息，请参阅《Amazon EKS 用户指南》中的创建 Amazon EKS 节点 IAM 角色和向 IAM 主体授予访问集群的权限。如果您使用的是容器组（pod）联网，请参阅《Amazon EKS 用户指南》中的创建适用于 Kubernetes 的 Amazon VPC CNI 插件以使用服务账户的 IAM 角色。

```
$ cat <<EOF > ./batch-eks-compute-environment.json
{
  "computeEnvironmentName": "My-Eks-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:region-code:your-account-number:cluster/my-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
```

备注

- 对 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。

第 4 步：创建作业队列并连接计算环境

> **Important**
>
> 在继续操作之前，请务必确认计算环境是否正常。DescribeComputeEnvironments API 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。

提交到这个新作业队列的作业在加入与您的计算环境关联的 Amazon EKS 集群的 AWS Batch 托管节点上以容器组（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 5：创建作业定义

以下作业定义会指示该容器组睡眠 60 秒。

```
$ 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
```

备注

- 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 6 步：提交作业

运行以下 AWS CLI 命令提交一个新作业。

```
$ aws batch submit-job --job-queue My-Eks-JQ1 \
    --job-definition MyJobOnEks_Sleep --job-name My-Eks-Job1
```

检查作业状态：

```
$ aws batch describe-jobs --job <jobId-from-submit-response>
```

备注

- 有关在 Amazon EKS 资源上运行作业的更多信息，请参阅 Amazon EKS 作业。

第 7 步：查看作业输出

要查看作业输出，请执行以下操作：

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择作业。
3. 在作业队列下拉列表中，选择您为本教程创建的作业队列。
4. 作业表会列出您的所有作业及其当前状态。作业的状态变为成功后，选择该作业的名称 *My-Eks-JQ1* 以查看作业详细信息。
5. 在详细信息窗格中，启动时间和停止时间应间隔一分钟。

第 8 步：（可选）提交带覆盖的作业

此作业会覆盖传递给容器的命令。AWS Batch 在作业完成后会积极清理容器组以减少 Kubernetes 的负载。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。

```
$ 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
```

备注

- 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。

要查看作业输出，请执行以下操作：

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择作业。
3. 在作业队列下拉列表中，选择您为本教程创建的作业队列。
4. 作业表会列出您的所有作业及其当前状态。作业的状态变为成功后，选择该作业的名称以查看该作业的详细信息。

5. 在详细信息窗格中，选择日志流名称。这时将会打开该作业的 CloudWatch 控制台，并且应会有一个消息为 `hello world` 或自定义消息的事件。

第 9 步：清理教程资源

启用 Amazon EC2 实例后，您需要为该实例付费。您可以删除实例以停止产生费用。

要删除您创建的资源，请执行以下操作：

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择作业队列。
3. 在作业队列表中，选择您为本教程创建的作业队列。
4. 选择 禁用。当作业队列的状态变为“已禁用”后，您可以选择删除。
5. 删除该作业队列后，在导航窗格中选择计算环境。
6. 选择您为本教程创建的计算环境，然后选择禁用。计算环境的禁用可能需要 1-2 分钟才能完成。
7. 计算环境的状态变为“已禁用”后，选择删除。计算环境的删除可能需要 1-2 分钟才能完成。

其他资源

完成本教程后，您可以探索以下主题：

- 详细了解最佳实践。
- 探索 AWS Batch 的核心组件。有关更多信息，请参阅 AWS Batch 的组成部分。
- 详细了解 AWS Batch 支持的各种计算环境。
- 详细了解作业队列及其各种调度选项。
- 详细了解作业定义以及各种配置选项。
- 详细了解各种不同的作业类型。

在 Amazon EKS 私有集群上使用 AWS Batch 的入门

AWS Batch 是一项托管服务，可编排 Amazon Elastic Kubernetes Service（Amazon EKS）集群中的批处理工作负载。该服务包括队列、依赖项跟踪、托管作业重试次数和优先级、容器组（pod）管理和节点扩展。此功能可将您现有的私有 Amazon EKS 集群与 AWS Batch 连接起来，从而大规模运行您

的作业。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 集群的 AWS Command Line Interface。

默认情况下，Amazon EKS 全私有集群没有入站/出站互联网访问权限，并且只能从 VPC 或已连接网络内访问 API 服务器。Amazon VPC 端点用于实现对其他 AWS 服务的私有访问。`eksctl` 支持使用预先存在的 Amazon VPC 和子网创建完全私有的集群。`eksctl` 还会在提供的 Amazon VPC 中创建 Amazon VPC 端点，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 Amazon VPC 中的容器注册表中拉取映像。同样，您可以在 Amazon VPC 中创建 Amazon Elastic Container Registry，并将容器映像复制到其中，以供节点拉取。有关更多信息，请参阅将容器映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 创建缓存提取规则。为外部公有注册表创建缓存提取规则后，就可以使用 Amazon ECR 私有注册表统一资源标识符（URI）从该外部公有注册表中提取映像。然后，Amazon ECR 会创建一个存储库并缓存映像。当使用 Amazon ECR 私有注册表 URI 提取缓存映像时，Amazon ECR 会检查远程注册表以查看是否有新版本的映像，并且会最多每 24 小时更新一次您的私有注册表。

目录

- 故障排除

概览

本教程展示了如何使用 AWS CloudShell、`kubectl` 和 `eksctl` 设置将 AWS Batch 用于一个私有 Amazon EKS。

目标受众

本教程面向负责设置、测试和部署 AWS Batch 的系统管理员和开发人员而设计。

使用的功能

本教程介绍如何使用 AWS CLI 来执行以下操作：

- 使用 Amazon Elastic Container Registry（Amazon ECR）存储容器映像
- 创建和配置 Amazon EKS 计算环境
- 创建作业队列。
- 创建任务定义
- 创建并提交要运行的作业
- 提交带覆盖的作业

所需时间

完成本教程大约需要 40–50 分钟。

区域限制

使用此解决方案没有任何国家或地区限制。

资源用量费用

创建 AWS 账户并不会收费；但是，通过实施此解决方案，您可能会产生下表中列出的部分或全部费用。

描述	费用（美元）
您需要按集群小时数付费	因实例而异，详情请参阅 Amazon EKS 定价
Amazon EC2 实例	您需要为创建的每个 Amazon EC2 实例付费。有关定价的更多信息，请参阅 Amazon EC2 定价。

先决条件

本教程使用的是 AWS CloudShell，这是一个已经事先完成身份验证的浏览器式 Shell，您可以直接从 AWS 管理控制台中启动。这样就可以在不再能够通过公共互联网访问集群时对其进行访问。AWS CLI、`kubectl`、和 `eksctl` 可能已经作为 AWS CloudShell 的一部分安装。有关 AWS CloudShell 的更多信息，请参阅《AWS CloudShell 用户指南》https://docs.aws.amazon.com/cloudshell/latest/userguide/welcome.html。一种替代 AWS CloudShell 的方法是连接到集群的 VPC 或集群的已连接网络。

要运行 kubectl 命令，您需要对 Amazon EKS 集群的私有访问权限。这意味着，传输到集群 API 服务器的所有流量都必须来自您的集群的 VPC 或连接的网络中。

- 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。在安装 AWS CLI 后，建议您还要对其进行配置。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的使用 `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`** – 用于处理 Amazon EKS 集群的命令行工具，该工具可自动执行许多单独任务。本指南要求您使用 `0.115.0` 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`。
- 权限：调用 CreateComputeEnvironment API 操作来创建要使用 Amazon EKS 资源的计算环境时，用户需要具有 `eks:DescribeCluster` 和 `eks:ListClusters` API 操作的权限。您可以按照《IAM 用户指南》中添加和删除 IAM 身份权限的说明，将 AWSBatchFullAccess 托管式策略附加到您的用户账户。
- 实例角色：您需要为您的 Amazon EKS 节点创建一个具有 `AmazonEKSWorkerNodePolicy` 和 `AmazonEC2ContainerRegistryPullOnly` 策略的 `InstanceRole`。有关如何创建 `InstanceRole` 的说明，请参阅创建 Amazon EKS 节点 IAM 角色。您将需要 `InstanceRole` 的 ARN。
- AWS 账户 ID：您需要知道您的 AWS 账户 ID。按照查看您的 AWS 账户 ID 中的说明进行操作。
- （可选）CloudWatch ：要检查（可选）提交带覆盖的作业的详细信息，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。

第 1 步：为 AWS Batch 创建 EKS 集群

> **Important**
>
> 本教程包含的步骤均使用默认设置，以助您尽可能简单快速地入门。在出于生产用途而开始创建之前，我们建议您熟悉所有设置，并使用符合您要求的设置进行部署。

我们建议您使用 `eksctl` 和以下配置文件来创建集群。要手动设置集群，请按照《Amazon EKS 用户指南》中部署具有有限互联网访问权限的私有集群部分的说明进操作。

1. 打开 AWS CloudShell 控制台并将区域设置为 `us-east-1`。在本教程的其余部分中，请确保您使用的是 `us-east-1`。
2. 使用示例 `eksctl` 配置文件，在 `us-east-1` 区域中创建一个私有 EKS 集群。将该 yaml 文件保存到您的 AWS CloudShell 环境中并将其命名为 `clusterConfig.yaml`。您可以将 *`my-test-cluster`* 更改为要使用的集群名称。

```
kind: ClusterConfig
apiVersion: eksctl.io/v1alpha5
metadata:
    name: my-test-cluster
    region: us-east-1
availabilityZones:
    - us-east-1a
    - us-east-1b
    - us-east-1c
managedNodeGroups:
    - name: ng-1
      privateNetworking: true
privateCluster:
    enabled: true
    skipEndpointCreation: false
```

3. 使用以下命令创建资源：`eksctl create cluster -f clusterConfig.yaml`。集群创建可能需要 10–15 分钟时间。
4. 集群创建完成后，您必须将您的 AWS CloudShell IP 地址添加到允许列表中。要查找您的 AWS CloudShell IP 地址，请运行以下命令：

```
curl http://checkip.amazonaws.com
```

获得公有 IP 地址后，您必须创建一条允许列表规则：

```
aws eks update-cluster-config \
  --name my-test-cluster \
  --region us-east-1 \
  --resources-vpc-config
 endpointPublicAccess=true,endpointPrivateAccess=true,publicAccessCidrs=["<Public
 IP>/32"]
```

然后将更新应用到该 kubectl 配置文件：

```
aws eks update-kubeconfig --name my-test-cluster --region us-east-1
```

5. 要测试您是否有权访问节点，请运行以下命令：

```
kubectl get nodes
```

命令的输出为：

```
NAME                                STATUS   ROLES    AGE     VERSION
ip-192-168-107-235.ec2.internal   Ready    none     1h   v1.32.3-eks-473151a
ip-192-168-165-40.ec2.internal    Ready    none     1h   v1.32.3-eks-473151a
ip-192-168-98-54.ec2.internal     Ready    none     1h   v1.32.1-eks-5d632ec
```

第 2 步：为 AWS Batch 准备好 EKS 集群

必须完成所有步骤，并且必须在 AWS CloudShell 中完成。

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。

```
$ 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: ["events"]
    verbs: ["list"]
  - 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 AWS Batch 创建名称空间范围的 Kubernetes 角色，用于管理和绑定生命周期容器组（pod）。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 Kubernetes aws-auth 配置映射以将前面的 RBAC 权限映射到 AWS Batch 服务相关角色。

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

输出：

```
2022-10-25 20:19:57 [#]  adding identity "arn:aws:iam::<your-account-ID>: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 aws-authconfigmap 中带路径的角色在其 ARN 中包含路径时不起作用。

第 3 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 用以管理您的 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。

由于 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。首先，创建一个指向 Amazon EKS 集群的计算环境。

- 对于 `subnets`，请运行 `eksctl get cluster` *`my-test-cluster`* 来获取集群使用的子网。
- 对于 `securityGroupIds` 参数，您可以使用与 Amazon EKS 集群相同的安全组。此命令检索集群的安全组 ID。

```
$ aws eks describe-cluster \
    --name my-test-cluster \
    --query cluster.resourcesVpcConfig.clusterSecurityGroupId
```

- 使用您在先决条件步骤中创建的 `instanceRole` ARN。

```
$ cat <<EOF > ./batch-eks-compute-environment.json
{
  "computeEnvironmentName": "My-Eks-CE1",
  "type": "MANAGED",
  "state": "ENABLED",
  "eksConfiguration": {
    "eksClusterArn": "arn:aws:eks:us-east-1:<your-account-ID>:cluster/my-test-cluster",
    "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
```

备注

- 对 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 中的安全性。

第 4 步：创建作业队列并连接计算环境

> **Important**
>
> 在继续操作之前，请务必确认计算环境是否正常。DescribeComputeEnvironments API 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。

提交到这个新作业队列的作业在加入与您的计算环境关联的 Amazon EKS 集群的 AWS Batch 托管节点上以容器组（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
```

第 5 步：创建具有缓存提取规则的 Amazon ECR

由于集群没有公共互联网问访权限，因此您必须为容器映像创建一个 Amazon ECR。以下说明会创建一个具有缓存提取规则的 Amazon ECR 来存储映像。

1. 以下命令会创建缓存提取规则。您可以使用其他前缀来替换 *tutorial-prefix*。

   ```
   aws ecr create-pull-through-cache-rule \
       --ecr-repository-prefix "my-prefix" \
       --upstream-registry-url "public.ecr.aws" \
       --region us-east-1
   ```

2. 完成公有 ECR 的身份验证。

   ```
   aws ecr get-login-password --region us-east-1 | docker login --username AWS --
   password-stdin <your-account-ID>.dkr.ecr.us-east-1.amazonaws.com
   ```

   现在就可以提取映像。

   ```
   docker pull <your-account-ID>.dkr.ecr.us-east-1.amazonaws.com/my-prefix/
   amazonlinux/amazonlinux:2
   ```

3. 您可以通过运行以下命令验证存储库和映像：

   ```
   aws ecr describe-repositories
   ```

```
aws ecr describe-images --repository-name my-prefix/amazonlinux/amazonlinux
```

4. 用于提取容器的映像字符串格式如下：

```
<your-account-ID>.dkr.ecr.us-east-1.amazonaws.com/my-prefix/amazonlinux/
amazonlinux:2
```

第 6 步：注册作业定义

以下作业定义会指示该容器组睡眠 60 秒。

在作业定义的映像字段中，与其提供指向公共 ECR 存储库中映像的链接，不如提供指向我们的私有 ECR 存储库中存储的映像的链接。请参阅以下示例作业定义：

```
$ cat <<EOF > ./batch-eks-job-definition.json
{
  "jobDefinitionName": "MyJobOnEks_Sleep",
  "type": "container",
  "eksProperties": {
    "podProperties": {
      "hostNetwork": true,
      "containers": [
        {
          "image": "<your-account-ID>.dkr.ecr.us-east-1.amazonaws.com/my-prefix/
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
```

备注

- 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 7 步：提交要运行的作业

在 AWS CloudShell 中运行以下 AWS CLI 命令，以提交一个新作业并返回唯一的 JobID。

```
$ aws batch submit-job --job-queue My-Eks-JQ1 \
    --job-definition MyJobOnEks_Sleep - -job-name My-Eks-Job1
```

备注

- 有关在 Amazon EKS 资源上运行作业的更多信息，请参阅 Amazon EKS 作业。

第 8 步：查看作业输出

检查作业状态：

```
$ aws batch describe-jobs --job <JobID-from-submit-response>
```

startedAt 和 stoppedAt 应会间隔一分钟。

第 9 步：（可选）提交带覆盖的作业

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
```

备注

- 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。

第 10 步：清理教程资源

启用 Amazon EC2 实例后，您需要为该实例付费。您可以删除实例以停止产生费用。

要删除您创建的资源，请执行以下操作：

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择作业队列。
3. 在作业队列表中，选择您为本教程创建的作业队列。

4. 对于操作，请选择禁用。当作业队列的状态变为“已禁用”后，您可以选择删除。
5. 删除该作业队列后，在导航窗格中选择计算环境。
6. 选择您为本教程创建的计算环境，然后在操作中选择禁用。计算环境的禁用可能需要 1-2 分钟才能完成。
7. 计算环境的状态变为“已禁用”后，选择删除。计算环境的删除可能需要 1-2 分钟才能完成。

其他资源

完成本教程后，您可以探索以下主题：

- 详细了解最佳实践。
- 探索 AWS Batch 的核心组件。有关更多信息，请参阅 AWS Batch 的组成部分。
- 详细了解 AWS Batch 支持的各种计算环境。
- 详细了解作业队列及其各种调度选项。
- 详细了解作业定义以及各种配置选项。
- 详细了解各种不同的作业类型。

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），则您的作业可能会保持“启动”状态。这是因为容器组（pod）将无法下载映像并运行您的 AWS Batch 作业。如果您点击 AWS Batch 启动的容器组（pod）名称，您应该能够看到错误消息并确认问题。错误消息应该类似于以下内容：

```
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，请参阅排查 AWS Batch 问题。有关基于容器组状态的问题排查，请参阅我如何排查 Amazon EKS 中的容器组（pod）状态问题？。

SageMaker 人工智能 AWS Batch 入门

AWS Batch 服务作业使您能够通过具有计划、优先级划分和排队功能的 AWS Batch 作业队列提交 SageMaker 训练作业。本教程演示如何使用 AWS Batch 服务作业设置和运行简单的 SageMaker 训练作业。

目录

概述

本教程演示如何使用为 SageMaker 训练作业设置 AWS Batch 服务作业 AWS CLI。

目标受众

本教程专为负责大规模设置和运行机器学习训练作业的数据科学家和开发人员设计。

使用的功能

本教程向您展示了如何使用 t AWS CLI o：

- 为 SageMaker 训练作业创建服务环境
- 创建 SageMaker 训练作业队列
- 使用 `SubmitServiceJob` API 提交服务作业
- 监控作业状态并查看输出
- 访问训练作业的 CloudWatch 日志

所需时间

完成本教程大约需要 15 分钟。

区域限制

本教程可以在任何同时提供 SageMaker AI AWS Batch 和 AI 的 AWS 地区完成。

资源使用成本

创建 AWS 账户不收取任何费用。但是，实施此解决方案可能需要为以下资源付费：

说明	费用（美元）
SageMaker AI 训练实例	您需要为使用的每个 SageMaker AI 训练实例付费。有关定价的更多信息，请参阅 SageMaker AI 定价。
Amazon S3 存储	存储训练作业输出的成本极低。有关更多信息，请参阅 Amazon S3 定价。

先决条件

在开始本教程之前，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 SageMaker AI 资源。

- AWS CLI— 用于处理 AWS 服务（包括 AWS Batch 和 SageMaker AI）的命令行工具。本指南要求您使用 2.8.6 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。

步骤 1：创建 A SageMaker I 执行角色

SageMaker AI 使用执行角色代表您使用其他 AWS 服务执行操作。您必须创建执行角色并授予 SageMaker AI 权限，才能使用训练作业所需的服务和资源。使用 `AmazonSageMakerFullAccess` 托管式策略，因为该策略包含了 Amazon S3 权限。

> **Note**
>
> 按照以下说明为本教程创建 SageMaker AI 执行角色。
> 在为生产环境创建执行角色之前，我们建议您查看《SageMaker AI 开发者指南》中的“如何使用 SageMaker AI 执行角色”。

1. 创建 IAM 角色

   使用以下信任策略创建名为 `sagemaker-trust-policy.json` 的 JSON 文件：

   JSON

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

   使用信任策略创建 IAM 角色：

   ```
   aws iam create-role \
       --role-name SageMakerExecutionRole \
       --assume-role-policy-document file://sagemaker-trust-policy.json \
       --description "Execution role for SageMaker training jobs"
   ```

2. 附加 托管式策略

   将所需的托管式策略附加到该角色：

   ```
   aws iam attach-role-policy \
       --role-name SageMakerExecutionRole \
   ```

```
    --policy-arn arn:aws:iam::aws:policy/AmazonSageMakerFullAccess
```

```
aws iam attach-role-policy \
    --role-name SageMakerExecutionRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonS3FullAccess
```

3. 记下角色 ARN

   获取角色 ARN，您将在后续步骤中使用它：

```
aws iam get-role --role-name SageMakerExecutionRole --query 'Role.Arn' --output
 text
```

   保存此 ARN，因为您将在创建训练作业有效载荷时使用它。

第 2 步：创建服务环境

服务环境定义了 SageMaker 训练作业的容量限制。服务环境包含了可以同时运行的最大训练实例数。

> **Important**
>
> 当您为 T SageMaker raining 创建第一个服务环境时，会在您的账户`AWSServiceRoleForAWSBatchWithSagemaker`中 AWS Batch 自动创建一个名为的服务相关角色。此角色 AWS Batch 允许您代表您对 SageMaker 培训作业进行排队和管理。有关此服务相关角色及其权限的更多信息，请参阅the section called “SageMaker 集成角色”。

创建一个最多可处理 5 个实例的服务环境：

```
aws batch create-service-environment \
    --service-environment-name TutorialServiceEnvironment \
    --service-environment-type SAGEMAKER_TRAINING \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=5
```

输出：

```
{
    "serviceEnvironmentName": "TutorialServiceEnvironment",
```

```
    "serviceEnvironmentArn": "arn:aws:batch:your-region:your-account-id:service-
environment/TutorialServiceEnvironment"
}
```

验证服务环境是否已成功创建：

```
aws batch describe-service-environments --service-
environments TutorialServiceEnvironment
```

输出：

```
{
    "serviceEnvironments": [
        {
            "serviceEnvironmentName": "TutorialServiceEnvironment",
            "serviceEnvironmentArn": "arn:aws:batch:your-region:your-account-
id:service-environment/TutorialServiceEnvironment",
            "serviceEnvironmentType": "SAGEMAKER_TRAINING",
            "state": "ENABLED",
            "status": "VALID",
            "capacityLimits": [
                {
                    "maxCapacity": 5,
                    "capacityUnit": "NUM_INSTANCES"
                }
            ],
            "tags": {}
        }
    ]
}
```

有关服务环境的更多信息，请参阅服务环境。

步骤 3：创建 SageMaker 任务队列

SageMaker 作业队列管理服务作业的调度和执行。提交到该队列的作业将根据可用容量分配到您的服务环境。

创建 SageMaker 训练作业队列：

```
aws batch create-job-queue \
```

```
    --job-queue-name my-sm-training-fifo-jq \
    --job-queue-type SAGEMAKER_TRAINING \
    --priority 1 \
    --service-environment-order order=1,serviceEnvironment=TutorialServiceEnvironment
```

输出：

```
{
    "jobQueueName": "my-sm-training-fifo-jq",
    "jobQueueArn": "arn:aws:batch:your-region:your-account-id:job-queue/my-sm-training-fifo-jq"
}
```

验证作业队列是否已成功创建：

```
aws batch describe-job-queues --job-queues my-sm-training-fifo-jq
```

输出：

```
{
    "jobQueues": [
        {
            "jobQueueName": "my-sm-training-fifo-jq",
            "jobQueueArn": "arn:aws:batch:your-region:your-account-id:job-queue/my-sm-training-fifo-jq",
            "state": "ENABLED",
            "status": "VALID",
            "statusReason": "JobQueue Healthy",
            "priority": 1,
            "computeEnvironmentOrder": [],
            "serviceEnvironmentOrder": [
                {
                    "order": 1,
                    "serviceEnvironment": "arn:aws:batch:your-region:your-account-id:service-environment/TutorialServiceEnvironment"
                }
            ],
            "jobQueueType": "SAGEMAKER_TRAINING",
            "tags": {}
        }
    ]
}
```

有关 SageMaker 任务队列的更多信息，请参阅the section called “创建 SageMaker 作业队列”。

第 4 步：创建和提交训练作业

现在，您将创建一个简单的训练作业并将其提交到您的作业队列中。此示例使用一个基本的“hello world”训练作业来演示服务作业的功能。

使用以下内容创建名为 *my_training_job.json* 的文件。*your-account-id*用您的 AWS 账户 ID 替换：

> **Note**
>
> S3OutputPath是创建 SageMaker 训练任务所必需的，但是本教程的结果未存储在 Amazon S3 存储桶中，您可以使用以下 JSON 中的路径。在生产环境中，如果您选择使用 Amazon S3 存储桶来存储输出，则需要有效的存储桶。

```
{
    "TrainingJobName": "my-simple-training-job",
    "RoleArn": "arn:aws:iam::your-account-id:role/SageMakerExecutionRole",
    "AlgorithmSpecification": {
        "TrainingInputMode": "File",
        "TrainingImage": "763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-
training:2.0.0-cpu-py310",
        "ContainerEntrypoint": [
            "echo",
            "hello world"
        ]
    },
    "ResourceConfig": {
        "InstanceType": "ml.c5.xlarge",
        "InstanceCount": 1,
        "VolumeSizeInGB": 1
    },
    "OutputDataConfig": {
        "S3OutputPath": "s3://your-s3-bucket/output"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 30
    }
}
```

使用 SubmitServiceJobAPI 提交训练作业：

```
aws batch submit-service-job \
    --job-queue my-sm-training-fifo-jq \
    --job-name my-batch-sm-job \
    --service-job-type SAGEMAKER_TRAINING \
    --retry-strategy attempts=1 \
    --timeout-config attemptDurationSeconds=60 \
    --service-request-payload file://my_training_job.json
```

输出：

```
{
    "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
    "jobName": "my-batch-sm-job",
    "jobId": "your-job-id"
}
```

有关服务作业有效载荷的更多信息，请参阅the section called “服务作业有效载荷”。有关提交服务作业的更多信息，请参阅the section called “提交服务作业”。

第 5 步：监控作业状态

您可以使用以下方法监控您的训练作业 AWS Batch APIs：DescribeServiceJobListServiceJobs、和GetJobQueueSnapshot。本节介绍检查作业状态和队列信息的不同方法。

查看队列中正在运行的作业：

```
aws batch list-service-jobs \
    --job-queue my-sm-training-fifo-jq --job-status RUNNING
```

输出：

```
{
    "jobSummaryList": [
        {
            "latestAttempt": {
                "serviceResourceId": {
                    "name": "TrainingJobArn",
                    "value": "arn:aws:sagemaker:your-region:your-account-id:training-
job/AWSBatch<my-simple-training-job><your-attempt-id>"
```

```
                }
            },
            "createdAt": 1753718760,
            "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
            "jobId": "your-job-id",
            "jobName": "my-batch-sm-job",
            "serviceJobType": "SAGEMAKER_TRAINING",
            "status": "RUNNING",
            "startedAt": 1753718820
        }
    ]
}
```

查看处于 RUNNABLE 状态的作业：

```
aws batch list-service-jobs \
    --job-queue my-sm-training-fifo-jq --job-status RUNNABLE
```

获取队列中即将执行的作业的快照：

```
aws batch get-job-queue-snapshot --job-queue my-sm-training-fifo-jq
```

输出：

```
{
    "frontOfQueue": {
        "jobs": [
            {
                "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
                "earliestTimeAtPosition": 1753718880
            },
            {
                "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id-2",
                "earliestTimeAtPosition": 1753718940
            }
        ],
        "lastUpdatedAt": 1753718970
    }
}
```

按名称搜索作业：

```
aws batch list-service-jobs \
    --job-queue my-sm-training-fifo-jq \
    --filters name=JOB_NAME,values="my-batch-sm-job"
```

输出：

```
{
    "jobSummaryList": [
        {
            "latestAttempt": {
                "serviceResourceId": {
                    "name": "TrainingJobArn",
                    "value": "arn:aws:sagemaker:your-region:your-account-id:training-
job/AWSBatch<my-simple-training-job><your-attempt-id>"
                }
            },
            "createdAt": 1753718760,
            "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-
id",
            "jobId": "your-job-id",
            "jobName": "my-batch-sm-job",
            "serviceJobType": "SAGEMAKER_TRAINING",
            "status": "RUNNING"
        }
    ]
}
```

有关作业状态映射的更多信息，请参阅the section called “服务作业状态”。

第 6 步：查看作业输出

任务完成后，您可以通过两者和 SageMaker AI 查看其输出 AWS Batch 和日志 APIs。

从 AWS Batch以下地址获取有关您的工作的详细信息：

```
aws batch describe-service-job \
    --job-id your-job-id
```

输出：

```
{
    "attempts": [
        {
            "serviceResourceId": {
                "name": "TrainingJobArn",
                "value": "arn:aws:sagemaker:your-region:your-account-id:training-job/
AWSBatch<my-simple-training-job><your-attempt-id>"
            },
            "startedAt": 1753718820,
            "stoppedAt": 1753718880,
            "statusReason": "Received status from SageMaker: Training job completed"
        }
    ],
    "createdAt": 1753718760,
    "jobArn": "arn:aws:batch:your-region:your-account-id:service-job/your-job-id",
    "jobId": "your-job-id",
    "jobName": "my-batch-sm-job",
    "jobQueue": "arn:aws:batch:your-region:your-account-id:job-queue/my-sm-training-
fifo-jq",
    "latestAttempt": {
        "serviceResourceId": {
            "name": "TrainingJobArn",
            "value": "arn:aws:sagemaker:your-region:your-account-id:training-job/
AWSBatch<my-simple-training-job><your-attempt-id>"
        }
    },
    "retryStrategy": {
        "attempts": 1,
        "evaluateOnExit": []
    },
    "serviceRequestPayload": "your-training-job-request-json",
    "serviceJobType": "SAGEMAKER_TRAINING",
    "startedAt": 1753718820,
    "status": "SUCCEEDED",
    "statusReason": "Received status from SageMaker: Training job completed",
    "stoppedAt": 1753718880,
    "tags": {},
    "timeoutConfig": {
        "attemptDurationSeconds": 60
    }
}
```

此命令返回全面的作业信息，包括 SageMaker 训练作业 ARN，您可以使用该信息直接通过 SageMaker AI 访问作业：

```
aws sagemaker describe-training-job \
    --training-job-name AWSBatch<my-simple-training-job><your-attempt-id>
```

要查看训练作业的 CloudWatch 日志，请先获取日志流名称：

```
aws logs describe-log-streams \
    --log-group-name /aws/sagemaker/TrainingJobs \
    --log-stream-name-prefix AWSBatchmy-simple-training-job
```

输出：

```
{
    "logStreams": [
        {
            "logStreamName": "your-log-stream-name",
            "creationTime": 1753718830,
            "firstEventTimestamp": 1753718840,
            "lastEventTimestamp": 1753718850,
            "lastIngestionTime": 1753718860,
            "uploadSequenceToken": upload-sequence-token,
            "arn": "arn:aws:logs:your-region:your-account-id:log-group:/aws/sagemaker/
TrainingJobs:log-stream:AWSBatch<my-simple-training-job><your-attempt-id>/algo-1-algo-
id",
            "storedBytes": 0
        }
    ]
}
```

然后使用上一响应中的日志流名称检索日志：

```
aws logs get-log-events \
    --log-group-name /aws/sagemaker/TrainingJobs \
    --log-stream-name your-log-stream-name
```

输出：

```
{
    "events": [
```

```
        {
            "timestamp": 1753718845,
            "message": "hello world",
            "ingestionTime": 1753718865
        }
    ],
    "nextForwardToken": "next-forward-token",
    "nextBackwardToken": "next-backward-token"
}
```

日志输出会显示来自训练作业的“hello world”消息，确认作业已成功执行。

第 7 步：清理教程资源

完成本教程后，请清除您创建的资源，以免持续产生费用。

首先，禁用并删除作业队列：

```
aws batch update-job-queue \
    --job-queue my-sm-training-fifo-jq \
    --state DISABLED
```

等待作业队列被禁用，然后将其删除：

```
aws batch delete-job-queue \
    --job-queue my-sm-training-fifo-jq
```

然后禁用并删除服务环境：

```
aws batch update-service-environment \
    --service-environment TutorialServiceEnvironment \
    --state DISABLED
```

等待服务环境被禁用，然后将其删除：

```
aws batch delete-service-environment \
    --service-environment TutorialServiceEnvironment
```

其他资源

完成本教程后，您可以探索以下主题：

- 我们建议使用 PySDK 创建服务作业并将其提交到您的作业队列，因为 PySDK 提供了帮助程序类和实用程序。有关使用 pySDK 的示例，请参阅上的 SageMaker AI 示例。 GitHub
- 了解有关 the section called “服务作业” 的更多信息。
- 探索 中的服务任务有效负载 AWS Batch，以了解更复杂的训练作业配置。
- 了解有关 在中提交服务作业 AWS Batch 和 `SubmitServiceJob` API 的信息。
- 参阅将 AWS Batch 服务作业状态映射到 SageMaker AI 状态以了解作业状态变换。
- 请访问 SageMaker AI Python SDK 文档，了解更多使用 Python 创建和提交 SageMaker 训练作业的功能丰富的方法。
- 浏览SageMaker示例笔记本，了解更复杂的机器学习工作流程。

AWS Batch 小组件控制面板

通过 AWS Batch 控制面板，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 的计算环境 AWS Batch。

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添加的其他小组件。

主题

如何将“单个作业队列”小组件添加到 AWS Batch 控制面板

“单个作业队列”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 管理控制台。
2. 从导航栏中，选择您需要的 AWS 区域。
3. 在导航窗格中，选择控制面板。
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选择添加。

如何将 CloudWatch Container Insights 小组件添加到 AWS Batch 控制面板

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 the section called “CloudWatch Container Insights”。

要添加此小组件，请按照以下步骤操作。

1. 打开 AWS Batch 管理控制台。
2. 从导航栏中，选择您需要的 AWS 区域。
3. 在导航窗格中，选择控制面板。
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. 选择添加。

如何将“作业日志”小组件添加到 AWS Batch 控制面板

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 “在 “日志” 中查看作业 CloudWatch 日志”。

要添加此小组件，请按照以下步骤操作。

1. 打开 AWS Batch 管理控制台。
2. 从导航栏中，选择您需要的 AWS 区域。
3. 在导航窗格中，选择控制面板。
4. 选择添加小部件。
5. 对于作业日志，选择添加小部件。
6. 在作业 ID 中，输入所需作业的作业 ID。
7. 选择添加。

的计算环境 AWS Batch

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

Important

AWS Batch 代表您并在您的账户中创建和管理多个 AWS 资源，包括亚马逊 EC2 启动模板、亚马逊 EC2 Auto Scaling 群组、亚马逊 EC2 竞价队列和亚马逊 ECS 集群。这些托管式资源均经过专门配置，以确保 AWS Batch 的最优运行状态。除非 AWS Batch 文档中明确说明，否则手动修改这些由 Batch 托管的资源可能会导致 `INVALID` 计算环境中的异常行为、不理想的实例扩缩行为、工作负载处理延迟，或产生意外的成本。该服务无法确定性地支持这些手动修改。 AWS Batch 请务必使用支持的 Batch APIs 或 Batch 控制台来管理您的计算环境。

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 创建后不会 AMIs 在计算环境中自动升级。例如，当更新版本的 Amazon ECS 优化的 AMI 发布时，它不会 AMIs 在您的计算环境中更新。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 在 AWS Batch 中更新计算环境。要使用计算环境的增强更新进行更新 AMIs，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。在创建新的托管计算环境时，如果您指定的minvCpu值大于零，则 AWS Batch 会创建一个仅用于非 MNP 任务的实例池。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，您可以管理自己的计算资源。 AWS Batch 支持 Amazon ECS 和 Amazon EKS 的非托管计算环境，允许您在利用 Batch 的作业计划功能的同时保持对基础设施的控制。

> **Note**
>
> AWS 非托管计算环境不支持 Fargate 资源。

非托管的 Amazon ECS 计算环境

对于非托管的 Amazon ECS 计算环境，您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; 规范和教程：创建计算资源 AMI。

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
```

非托管的 Amazon EKS 计算环境

在非托管的 Amazon EKS 计算环境中，您可以管理自己的 Kubernetes 节点，同时 AWS Batch 处理任务调度和放置。它允许您直接控制您的 Kubernetes 基础架构，以满足安全性、合规性或运营要求。您负责配置和配置您的 Amazon EKS 节点，同时与现有的 Amazon EKS 集群 AWS Batch 集成以安排和运行作业。

有关更多信息，请参阅教程：使用 Amazon EKS 资源创建非托管计算环境。

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：

- 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。

主题

教程：使用 Fargate 资源创建托管计算环境

完成以下步骤，以使用 AWS Fargate 资源创建托管式计算环境。

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 将服务关联角色用于 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 BatchFargate 上的 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。

教程：使用 Amazon EC2 资源创建托管计算环境

完成以下步骤，以使用 Amazon Elastic Compute Cloud（Amazon EC2）资源创建托管式计算环境。

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，选择环境。
4. 选择创建环境，然后选择计算环境。
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. 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的 AWS API 操作。例如，选择 AWSServiceRoleForBatch。有关更多信息，请参阅 将服务关联角色用于 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。该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的 AWS 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。

      i. （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）。

      ii. （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 标记 AWS Batch 资源。

g. 选择下一步。

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**
>
> 要在创建时标记竞价型实例，您的 Amazon EC2 竞价型实例集 IAM 角色必须使用较新的 AmazonEC2SpotFleetTaggingRole 托管策略。AmazonEC2SpotFleetRole 托管策略不具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 “标记资源”。

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`。

如果您选择了以下选项之一，则 AWS Batch 可为您选择实例类型：

- `optimal`，以选择符合作业队列需求的实例类型（从 `c4`、`m4`、`r4`、`c5`、`m5` 和 `r5` 实例系列中选择）。

- default_x86_64 会选择符合作业队列的资源需求的基于 x86 的实例类型（从 m6i、c6i、r6i 和 c7i 实例系列中选择）。
- default_arm64 会选择符合作业队列的资源需求的基于 x86 的实例类型（从 m6g、c6g、r6g 和 c7g 实例系列中选择）。

> **Note**
>
> 从 2025 年 11 月 1 日起，optimal 的行为将更改为匹配 default_x86_64。在更改期间，您的实例系列可能会更新到新一代实例。本次升级无需您执行任何操作。有关更改的更多信息，请参阅
>
> > **Note**
> >
> > 从 2025 年 11 月 1 日起，optimal 的行为将更改为匹配 default_x86_64。在更改期间，您的实例系列可能会更新到新一代实例。本次升级无需您执行任何操作。
>
> AWS Batch 支持 Instan ceTypes 中的单个选项optimal来满足任务队列的需求。我们引入了两个新的实例类型选项：default_x86_64 和 default_arm64。如果您未选择任何实例类型，我们将使用 default_x86_64。启用这些新选项后，系统将根据作业队列的要求自动跨系列和代系选择经济实惠的实例类型，让工作负载能够快速运行。
>
> 当新的实例类型有足够的容量可用时 AWS 区域，相应的默认池将自动使用新的实例类型进行更新。现有的 optimal 选项将继续受支持，不会弃用，因为底层默认池将支持该选项，以在未来提供更新后的实例。如果您使用的是 optimal 选项，则无需执行任何操作。
>
> 但是，请注意，只有ENABLEDVALID计算环境 (CEs) 会自动使用新的实例类型进行更新。如果您有DISABLED或 INVALID CEs，则它们将在重新启用并设置为VALID状态后收到更新。
>
> 。

> **Note**
>
> - 实例系列的可用性因 AWS 区域而异。例如，有些 AWS 区域可能不提供任何第四代实例系列，而提供了第五代和第六代实例系列。
> - 使用 `default_x86_64` 或 `default_arm64` 实例捆绑包时，AWS Batch 将在综合权衡成本效益和性能的基础上选择实例系列。虽然新一代实例通常具有更好的性价比，但如果对于您的工作负载而言，较早一代实例系列提供最佳的可用性、成本和性能组合，则 AWS Batch 可能会选择较早一代的实例系列。例如，如果某个 AWS 区域同时提供 c6i 和 c7i 实例，而 c6i 实例对于您的具体作业要求而言具有更好的成本效益，则 AWS Batch 可能会选择 c6i 实例。有关 AWS Batch 实例类型和 AWS 区域可用性的更多信息，请参阅实例类型计算表。
> - AWS Batch 会定期将默认捆绑包中的实例更新为更具成本效益的新一代实例。这些更新会自动完成，无需您执行任何操作。您的工作负载会在更新期间继续运行，不会出现中断。

> **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 将根据您的作业队列中所需的数量扩展 GPU。要使用 GPU 调度功能，计算环境包含的实例类型必须为 p6, p3, p4, p5, g3, g3s, g4、g5 或 g6 系列。

h. 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 “实例类型分配策略”。

i. 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。

ii. （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。

iii. （可选）对于 EC2 配置，请选择映像类型和映像 ID 覆盖值，以向 AWS Batch 提供为计算环境中的实例选择亚马逊机器映像（AMI）所需的信息。如果未为每种映像类型指定映像 ID 覆盖，则 AWS Batch 选择最近经 Amazon ECS 优化的 AMI。如果未指定映像类型，则非 GPU、非 AWS Graviton 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

所有基于 AWS Graviton 的实例系列的默认值（例如 C6g、M6g、R6g 和 T4g），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），可用于所有非基于 AWS Graviton 的实例类型。

Amazon Linux 2023

AWS Batch 支持 Amazon Linux 2023。

> **Note**
>
> Amazon 2023 不支持 A1 实例。

Amazon Linux 2023 (GPU)

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

> **Note**
>
> 您为计算环境选择的 AMI 必须与您打算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 Amazon ECS 优化型 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。

j. （可选）展开启动模板

i. 对于默认启动模板，选择一个用来配置计算资源的现有 Amazon EC2 启动模板。模板的默认版本会自动填充。有关更多信息，请参阅 使用 Amazon EC2 启动模板和 AWS Batch。

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

ii. （可选）对于默认版本，输入 `$Default`、`$Latest` 或要使用的特定版本号。

> **Note**
>
> 注意：如果您使用替换变量（$Default 或 $Latest），则会在保存此配置时应用当前的默认或最新版本号。如果未来默认版本或最新版本发生更改，则必须更新相关信息——它不会自动更新。

> **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 “基础设施更新期间的 AMI 选择”。

iii. （可选）对于覆盖启动模板，选择添加覆盖启动模板

A. （可选）对于启动模板，选择一个要用于特定实例类型和系列的现有 Amazon EC2 启动模板。

B. （可选）对于默认版本，输入要使用的特定版本号、`$Default` 或 `$Latest`。

> **Note**
>
> 如果您使用 `$Default` 或 `$Latest` 变量，则 AWS Batch 将在创建计算环境时应用当前信息。如果未来默认版本或最新版本发生更改，则必须通过 UpdateComputeEnvironment 或通过 AWS 管理控制台 - AWS Batch.更新相关信息。

C. （可选）对于目标实例类型，选择要应用该覆盖启动模板的实例类型或系列。

> **Note**
>
> 如果您指定了覆盖启动模板，则必须指定目标实例类型。有关更多信息，请参阅 LaunchTemplateSpecificationOverride.targetInstanceTypes。

> **Note**
>
> 如果要选择的实例类型或系列未出现在此列表中，请检查您在 `Allowed instance types` 中所做的选择。

k. 选择下一步。

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、《Amazon Elastic Container Service 开发人员指南》中的本地区域、Wavelength Zones 和 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。

> **Note**
>
> 注意：如果您使用替换变量（$Default 或 $Latest），则会在保存此配置时应用当前的默认或最新版本号。如果未来默认版本或最新版本发生更改，则必须更新相关信息——它不会自动更新。

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。

教程：使用 Amazon EC2 资源创建非托管计算环境

完成以下步骤以使用 Amazon Elastic Compute Cloud（Amazon EC2）资源创建非托管式计算环境。

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. 对于服务角色，请选择一个角色，该角色允许 AWS Batch 服务代表您调用所需的 AWS API 操作。

> **Note**
>
> 您不能将 `AWSServiceRoleForBatch` 用于非托管计算环境。

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 标记 AWS Batch 资源。
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。

教程：使用 Amazon EKS 资源创建托管计算环境

完成以下步骤，以使用 Amazon Elastic Kubernetes Service（Amazon EKS）资源创建托管式计算环境。

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。

    如果您选择了以下选项之一，则 AWS Batch 可为您选择实例类型：

    - optimal，以选择符合作业队列需求的实例类型（从 c4、m4、r4、c5、m5 和 r5 实例系列中选择）。
    - default_x86_64 会选择符合作业队列的资源需求的基于 x86 的实例类型（从 m6i、c6i、r6i 和 c7i 实例系列中选择）。
    - default_arm64 会选择符合作业队列的资源需求的基于 x86 的实例类型（从 m6g、c6g、r6g 和 c7g 实例系列中选择）。

> **Note**
>
> 从 2025 年 11 月 1 日起，`optimal` 的行为将更改为匹配 `default_x86_64`。在更改期间，您的实例系列可能会更新到新一代实例。本次升级无需您执行任何操作。有关更改的更多信息，请参阅用于接收自动实例系列更新的最佳实例类型配置。

> **Note**
>
> - 实例系列的可用性因 AWS 区域而异。例如，有些 AWS 区域可能不提供任何第四代实例系列，而提供了第五代和第六代实例系列。
> - 使用 `default_x86_64` 或 `default_arm64` 实例捆绑包时，AWS Batch 将在综合权衡成本效益和性能的基础上选择实例系列。虽然新一代实例通常具有更好的性价比，但如果对于您的工作负载而言，较早一代实例系列提供最佳的可用性、成本和性能组合，则 AWS Batch 可能会选择较早一代的实例系列。例如，如果某个 AWS 区域同时提供 c6i 和 c7i 实例，而 c6i 实例对于您的具体作业要求而言具有更好的成本效益，则 AWS Batch 可能会选择 c6i 实例。有关 AWS Batch 实例类型和 AWS 区域可用性的更多信息，请参阅实例类型计算表。
> - AWS Batch 会定期将默认捆绑包中的实例更新为更具成本效益的新一代实例。这些更新会自动完成，无需您执行任何操作。您的工作负载会在更新期间继续运行，不会出现中断

> **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 将根据您的作业队列中所需的数量扩展 GPU。要使用 GPU 调度功能，计算环境包含的实例类型必须为 p6, p3, p4, p5, g3, g3s, g4、g5 或 g6 系列。

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。

       您可以使用 Amazon EKS 优化型 Amazon Linux AMI，也可以使用自定义 AMI。

       i. 使用 Amazon EKS 优化型 Amazon Linux AMI：

          A. 对于映像类型，请选择下列选项之一：

             - Amazon Linux 2：所有基于 AWS Graviton 的实例系列（例如 C6g、M6g、R6g 和 T4g）的默认值，可用于所有非 GPU 实例类型。
             - Amazon Linux 2（加速版）：所有 GPU 实例系列（例如 P4 和 G4）的默认值，可用于所有非基于 AWS Graviton 的实例类型。
             - Amazon Linux 2023：AWS Batch 支持 Amazon Linux 2023（AL2023）。
             - Amazon Linux 2023（加速型）：GPU 实例系列，可用于所有非基于 AWS Graviton 的实例类型。

          B. 对于 Kubernetes 版本，输入一个 Kubernetes 版本号。

       ii. 使用自定义 AMI：

          A. 对于映像类型，选择自定义 AMI 所基于的 AMI 类型：

             - Amazon Linux 2：所有基于 AWS Graviton 的实例系列（例如 C6g、M6g、R6g 和 T4g）的默认值，可用于所有非 GPU 实例类型。
             - Amazon Linux 2（加速版）：所有 GPU 实例系列（例如 P4 和 G4）的默认值，可用于所有非基于 AWS Graviton 的实例类型。
             - Amazon Linux 2023：AWS Batch 支持 AL2023。
             - Amazon Linux 2023（加速型）：GPU 实例系列，可用于所有非基于 AWS Graviton 的实例类型。

          B. 对于映像 ID 覆盖，输入自定义 AMI ID。

          C. 对于 Kubernetes 版本，输入一个 Kubernetes 版本号。

    d. （可选）对于启动模板，选择一个现有的启动模板。

    e. （可选）对于启动模板版本，输入 **$Default**、**$Latest** 或版本号码。

    f. （可选）对于覆盖启动模板，选择添加覆盖启动模板以添加覆盖：

i. （可选）对于启动模板，选择要添加该覆盖的启动模板。

ii. （可选）对于启动模板版本，选择启动模板的版本号、$Default 或 $Latest。

iii. （可选）对于目标实例类型，选择将应用此覆盖的实例类型或系列。这只能是允许的实例类型中包含的实例类型和系列。

iv. （可选）对于 userdataType，选择 EKS 节点初始化。只有在启动模板中指定了 AMI 或将其作为启动模板覆盖时，才使用此字段。对于基于 EKS_AL2023 或 EKS_AL2023_NVIDIA 的自定义 AMI，请选择 EKS_NODEADM；对于 EKS_AL2 和 EKS_AL_NVIDIA，请选择 EKS_BOOSTRAP_SH。默认值为 EKS_BOOSTRAP_SH。

对于在同一计算环境中同时使用基于 AL2 和基于 AL2023 的自定义 AMI 的混合环境，您需要使用 userdataType。

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 Amazon EKS 上支持 Local Zones。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS 以及 AWS Local 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。

教程：使用 Amazon EKS 资源创建非托管计算环境

完成以下步骤，使用亚马逊 Elastic Kubernetes Service (Amazon EKS) 资源创建非托管计算环境。

1. 打开 AWS Batch 控制台，网址为https://console.aws.amazon.com/batch/。
2. 从页面顶部的导航栏中，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. 对于计算环境的配置，选择 Amazon Elastic Kubernetes Service (Amazon EKS)。

   b. 对于编排类型，请选择非托管。

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 EKS 集群，选择现有的 Amazon EKS 集群。要创建新的 EKS 集群，请按照创建 Amazon EKS 集群页面上的步骤进行操作。

8. 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。

9. （可选）在 Maximum v 中CPUs，指定预配置容量中CPUs 可用于作业调度的最大 v 数。

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. 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。

> **将 Amazon EKS 集群节点分配给非托管计算环境**
>
> 创建非托管计算环境后，您需要使用计算环境 UUID 标记您的 Amazon EKS 节点。
> 首先，从 `DescribeComputeEnvironments` API 结果中获取计算环境 UUID：
>
> ```
> $ aws batch describe-compute-environments \
>     --compute-environments unmanagedEksCE \
>     --query "computeEnvironments[].{name: computeEnvironmentName, uuid: uuid}"
> ```
>
> 获取节点信息：
>
> ```
> kubectl get nodes -o name
> ```
>
> 使用 AWS Batch 计算环境 UUID 标记节点：
>
> ```
> kubectl label <node-name> batch.amazonaws.com/compute-environment-uuid=uuid
> ```

资源：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。

> **Note**
>
> 您可以使用以下 AWS CLI 命令生成计算环境模板。
>
> ```
> $ aws batch create-compute-environment --generate-cli-skeleton
> ```

```
{
    "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": ""
    }
}
```

实例类型计算表

下表列出了 AWS 区域、实例系列关键字和可用的实例系列。AWS Batch 将尝试分配最新系列的实例，但由于实例系列的可用性因 AWS 区域而异，因此您可能会获得较早代系的实例系列。

default_x86_64

区域	实例系列
所有支持 AWS Batch 的 AWS 区域	m6i、c6i、r6i c7i

default_arm64

区域	实例系列
所有支持 AWS Batch 的 AWS 区域	m6g、c6g、r6g

区域	实例系列
	c7g

最佳选择

区域	实例系列
• ap-northeast-1 • ap-northeast-2 • ap-south-1 • ap-southeast-1 • ap-southeast-2 • ca-central-1 • cn-north-1 • cn-northwest-1 • eu-central-1 • eu-west-1 • eu-west-2 • sa-east-1 • us-east-1 • us-east-2 • us-gov-west-1 • us-west-1 • us-west-2	m4、c4、r4
• af-south-1 • ap-east-1 • ap-northeast-3 • ap-south-2 • ap-southeast-3 • ap-southeast-4 • ca-west-1	m5、c5、r5

区域	实例系列
• eu-central-2 • eu-north-1 • eu-south-1 • eu-south-2 • eu-west-3 • il-central-1 • me-central-1 • me-south-1 • us-gov-east-1 • us-isob-east-1 • us-iso-east-1 • us-isof-south-1 • us-isof-east-1 • eu-isoe-west-1 • us-northeast-1	
• ap-southeast-5 • ap-southeast-7 • ap-east-2 • mx-central-1	m6、c6、r6

在 AWS Batch 中更新计算环境

AWS Batch 提供了多种计算环境更新策略，每种策略都针对特定的更新场景和要求而设计。这些方法使用相同的底层更新 API，但代表了用于有效管理更新的不同规范方法。您可以使用 AWS Batch 控制台或 AWS CLI 来管理这些更新。了解这些策略有助您选择最适合自己需求的方法，同时尽可能减少工作负载中断。

本主题概述了可用的更新策略，并提供了有关每种方法的使用场景的指导。有关详细操作过程，请参阅每种更新策略的相应部分。

> **Important**
>
> AWS Batch 会在您的账户中代表您创建和管理多个 AWS 资源，包括 Amazon EC2 启动模板、Amazon EC2 Auto Scaling 组、Amazon EC2 竞价型实例集和 Amazon ECS 集群等。这些托管式资源已特别为了确保最优的 AWS Batch 操作而进行配置。除非 AWS Batch 文档中明确说明，否则手动修改这些 AWS Batch 托管式资源可能会导致异常行为（包括 `INVALID` 计算环境）、不理想的实例扩缩行为、工作负载处理延迟，或产生意外的成本。不能确定 AWS Batch 服务一定能够支持这些手动修改。请务必使用支持的 AWS Batch API 或 AWS Batch 控制台来管理您的计算环境。

主题

计算环境更新策略

使用扩缩更新或基础设施更新策略时，您的计算环境将就地更新。对于蓝绿更新策略，您需要创建一个新计算环境（绿色），然后将工作负载从旧计算环境（蓝色）迁移到新计算环境（绿色）。

AWS Batch 提供了三种不同的计算环境更新策略：

扩缩更新

扩缩更新通过增加或移除实例来调整计算环境的容量，但不会替换现有实例。这是最快的更新场景，不需要停机时间。需要更改容量设置（vCPU 数）时应使用扩缩更新。此类更新通常会在几分钟内完成。

Fargate 更新的执行过程与扩缩更新相同。有关更多信息，请参阅 执行扩缩更新。

基础架构更新

基础设施更新会将计算环境中的实例替换为具有更新后设置的新实例。此类更新需要特定的服务角色和分配策略配置，但停机时间极少，正在运行的作业可能会出现中断。需要修改实例类型、AMI

配置、联网设置、服务角色、环境状态或其他基础设施组件时，应使用基础设施更新。根据作业完成情况，此类更新通常会在 10-30 分钟内完成。

有关更多信息，请参阅 执行基础设施更新。

蓝绿更新

蓝绿更新会创建一个与现有环境并行运行的新计算环境，从而实现在零停机时间的情况下逐步转移工作负载。这种方法提供了最安全的更新路径，但需要临时运行两个环境。需要零停机时间、希望在完全部署之前测试更改、需要快速回滚功能或使用不受支持的配置进行基础设施更新时，应使用蓝绿更新。完成更新所需的时间因情况而异，并且由您控制。

有关更多信息，请参阅 为计算环境执行蓝绿更新。

选择合适的更新策略

使用以下决策指南来选择最适合您需求的更新策略：

选择扩缩更新的场景

只需要调整计算容量（vCPU 数）时，应选择扩缩更新策略。需要在无停机时间的情况下快速更新且无需更改基础设施配置时，扩缩更新是理想的选择。

有关详细步骤，请参阅 执行扩缩更新。

选择基础设施更新的场景

需要修改实例类型、AMI 设置、服务角色、环境状态或联网配置时，应选择基础设施更新策略。您的环境必须使用 AWSServiceRoleForBatch 服务相关角色以及 BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED 或 SPOT_PRICE_CAPACITY_OPTIMIZED 分配策略。如果更新期间可以接受某些作业中断，并且您希望自动更新到最新版本的 Amazon ECS 优化型 AMI，则非常适合选择基础设施更新。

有关详细步骤，请参阅 执行基础设施更新。

选择蓝绿更新的场景

如果工作负载需要零停机时间，或者您需要在转移生产工作负载之前测试更改，应选择蓝绿更新策略。当快速回滚功能很重要、您的环境使用 BEST_FIT 分配策略或者您的环境不使用 AWSServiceRoleForBatch 服务相关角色时，必须使用这种方法。使用需要手动更新或需要进行重大配置更改的自定义 AMI 时，蓝绿更新也是最佳选择。

有关详细步骤，请参阅 为计算环境执行蓝绿更新。

AMI 更新注意事项

当满足以下所有条件时，AWS Batch 可以在基础设施更新期间更新到最新版本的 Amazon ECS 优化型 AMI：

> **Note**
>
> 基础设施更新完成后，updateToLatestImageVersion 将设置为 false。要启动其他更新，必须将 updateToLatestImageVersion 设置为 true。

- 计算环境使用 AWSServiceRoleForBatch 服务相关角色
- 分配策略设置为 BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED 或 SPOT_PRICE_CAPACITY_OPTIMIZED
- 未在 imageId、imageIdOverride 或启动模板中显式指定 AMI ID
- updateToLatestImageVersion 设置为 true

使用蓝绿部署进行 AMI 更新

对于下列场景，您必须使用蓝绿部署来更新 AMI：

- 使用指定版本的 Amazon ECS 优化型 AMI 时
- 在以下任一资源中指定了 AMI ID 时：
  - 启动模板（必须更新模板或将其移除）
  - imageId 参数
  - EC2 配置中的 imageIdOverride 参数
- 使用 BEST_FIT 分配策略时（不支持基础设施更新）
- 不使用 AWSServiceRoleForBatch 服务相关角色时

执行扩缩更新

扩缩更新通过增加或移除实例来调整计算环境的容量。这是最快的更新策略，不需要替换现有实例。扩缩更新适用于任何服务角色类型和分配策略，因此是最灵活的更新选项。

触发扩缩更新的更改

当您仅修改以下设置时，AWS Batch 将执行扩缩更新。如果您修改了以下任何设置以及其他计算环境设置，AWS Batch 会改为执行基础设施更新。

仅修改以下设置时会触发扩缩更新：

- `desiredvCpus`：设置环境的目标 vCPU 数。
- `maxvCpus`：定义可启动的最大 vCPU 数。
- `minvCpus`：指定要维持的最小 vCPU 数。

对于 Fargate 计算环境，您还可以修改以下设置以执行扩缩更新：

- `securityGroupIds`：计算环境的安全组 ID。
- `subnets`：计算环境的子网。

> **Note**
>
> 我们建议不要使用 `desiredvCpus` 来启动扩缩更新，因为 AWS Batch 会动态调整 `desiredvCpus`。而应使用 `minvCpus`。
> 更新 `desiredvCpus` 时，该值必须介于 `minvCpus` 和 `maxvCpus` 之间。新值必须大于或等于当前的 `desiredvCpus`。有关更多信息，请参阅 the section called “更新desiredvCpus设置时出现错误消息”。

> **Important**
>
> 如果您修改以下任何一个扩缩设置以及其他计算环境设置（例如实例类型、AMI ID 或启动模板）一起修改，则 AWS Batch 会执行基础设施更新而不是扩缩更新。基础设施更新需要更长的时间，并且可能会替换现有实例。

Performing scaling updates using the AWS 管理控制台

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择环境，然后选择计算环境选项卡。

3. 选择要更新的计算环境。
4. 选择操作，然后选择编辑。
5. 修改一个或多个支持扩缩更新的设置。例如：
   - 对于最小 vCPU 数，输入最小 vCPU 数量。
   - 对于所需 vCPU 数，输入所需的 vCPU 数量。
   - 对于最大 vCPU 数，输入最大 vCPU 数量。
6. 选择保存更改。
7. 监控计算环境状态。更新应该很快完成，因为只涉及扩缩操作。

Performing scaling updates using the AWS CLI

使用 update-compute-environment 命令执行扩缩更新。以下两个示例展示了常见的扩缩操作：您可以修改以下一个或多个支持扩缩更新的设置

- 此示例会更新所需 vCPU 数、最小 vCPU 数和最大 vCPU 数：

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources minvCpus=2,maxvCpus=8
```

监控扩缩更新

使用 AWS Batch 控制台监控扩缩更新，以查看计算环境状态并检查实例数量和 vCPU 指标。您还可以通过 AWS CLI 使用 describe-compute-environments 命令来检查状态并监控实例计数和 vCPU 值。

执行基础设施更新

基础设施更新会将计算环境中的实例替换为具有更新后设置的新实例。此更新策略需要的时间比扩缩更新更长，并且需要特定的服务角色和分配策略设置。基础设施更新是一种在保持服务可用性的同时修改基本计算环境配置的方法。

> **Important**
>
> 基础设施更新需要 AWSServiceRoleForBatch 服务相关角色以及 BEST_FIT_PROGRESSIVE、SPOT_CAPACITY_OPTIMIZED 或

SPOT_PRICE_CAPACITY_OPTIMIZED 分配策略。如果您的环境不符合这些要求，请改用蓝绿更新。

触发基础设施更新的更改

当您修改以下任何设置时，AWS Batch 将执行基础设施更新。当您同时修改这些设置以及扩缩更新设置时，也会执行基础设施更新。

以下设置会触发基础设施更新：

计算配置

- `allocationStrategy`：确定 AWS Batch 将如何选择实例类型。
- `instanceTypes`：指定要使用的 EC2 实例类型。
- `bidPercentage`：竞价型实例的最大按需价格百分比。
- `type`：计算环境类型（EC2 或 SPOT）。

AMI 和启动配置

- `imageId`：要用于实例的特定 AMI。
- `ec2Configuration`：EC2 配置，包括 `imageIdOverride`。
- `launchTemplate`：EC2 启动模板设置。
- `ec2KeyPair`：用于实例访问的 SSH 密钥对。
- `updateToLatestImageVersion`：AMI 自动更新设置。

网络和安全性

- `subnets`：所启动的实例的 VPC 子网（适用于 EC2 计算环境）。
- `securityGroupIds`：实例的安全组（适用于 EC2 计算环境）。
- `placementGroup`：EC2 置放群组配置。

其他设置

- `instanceRole`：用于 EC2 实例的 IAM 角色。

- tags：应用到 EC2 实例的标签。

> **Important**
>
> 如果您修改了任何基础设施更新设置以及扩缩更新设置（例如 desiredvCpus、maxvCpus、或 minvCpus），则 AWS Batch 会执行基础设施更新。基础设施更新需要是时间比扩缩更新更长。

基础设施更新期间的 AMI 选择

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`。

更新期间的作业处理

使用更新策略配置在基础设施更新期间如何处理正在运行的作业。如果设置为 `terminateJobsOnUpdate=true`，则正在运行的作业将立即终止，`jobExecutionTimeoutMinutes` 设置将被忽略，并且更新将在可以替换实例时立即继续执行。如果设置为 `terminateJobsOnUpdate=false`，则正在运行的作业将在指定的超时期限内继续运行，默认超时期限为 30 分钟，超过超时期限的作业将被终止。

> **Note**
>
> 要重试在更新期间被终止的作业，请配置作业重试策略。有关更多信息，请参阅 the section called “自动作业重试”。

Performing infrastructure updates using the AWS 管理控制台

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择环境，然后选择计算环境选项卡。
3. 选择要更新的计算环境。
4. 选择操作，然后选择编辑。
5. 在更新行为部分中，配置如何处理正在运行的作业：
   - 选择将 AMI 更新到最新版本，以将 AMI 更新为最新版本。
   - 选择更新时立即终止作业，以在更新进程运行时终止作业。
   - 对于作业执行超时，输入在更新进程启动之前要等待的分钟数。
6. 修改一项或多项要求使用基础设施更新的设置。例如：
   - 实例角色
   - 使用 EC2 竞价型实例
   - 允许的实例类型
   - 置放群组 ()
   - EC2 密钥对

- EC2 配置 ()
- 启动模板
- 子网
- 安全组。

7. 选择保存更改。
8. 监控计算环境状态。环境将在更新过程中显示 UPDATING 状态。

Performing infrastructure updates using the AWS CLI

使用 update-compute-environment 命令并更改一项或多项要求使用基础设施更新的设置。以下三个示例是常见的基础设施操作。

- 此示例会更新实例类型并配置更新策略：

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources instanceTypes=default_x86_64 \
    --update-policy terminateJobsOnUpdate=false,jobExecutionTimeoutMinutes=30
```

- 此示例会更新 VPC 子网和安全组：

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources subnets=subnet-abcd1234,subnet-efgh5678
 securityGroupIds=sg-abcd1234 \
    --update-policy terminateJobsOnUpdate=true
```

- 此示例会自动更新到最新版本的 Amazon ECS 优化型 AMI：

```
aws batch update-compute-environment \
    --compute-environment your-compute-environment-name \
    --compute-resources updateToLatestImageVersion=true \
    --update-policy terminateJobsOnUpdate=false,jobExecutionTimeoutMinutes=60
```

监控基础设施更新

使用 AWS Batch 控制台监控基础设施更新，以观察计算环境的状态变为 UPDATING，监控实例替换进度，并检查是否有任何失败的更新。一旦计算环境状态变为 VAILD，即表示更新成功。您还可以使用

CloudWatch 来跟踪实例终止事件并在更新期间监控作业状态。使用 AWS CLI 时，请使用 describe-compute-environments 命令来检查状态和监控实例生命周期事件。

为计算环境执行蓝绿更新

蓝绿更新策略通过创建与现有计算环境（蓝色）并行运行的新计算环境（绿色），来减少停机时间和风险。借助这种方法，您可以在保持现有环境正常运的行同时，逐步将工作负载转移到新环境。蓝绿更新提供了最安全的更新路径，并且适用于任何服务角色类型或分配策略。

概览

蓝绿更新具有多项优势，是生产环境更新的理想方式。在更新过程中，您的工作负载会继续运行，从而实现零停机时间。这种方法支持轻松回滚功能，让您能够在出现问题时快速恢复到原始环境。您可以实施逐步转移策略，在生产工作负载完全割接之前验证新环境的性能。使用这种方法时，原始环境将保持不变并可以正常运行，直到您选择将其移除为止，因此还具有极佳的风险缓解作用。

何时需要使用蓝绿更新

在下列情况下必须使用蓝绿更新：

- 当您的计算环境使用 BEST_FIT 分配策略时（不支持基础设施更新）
- 当您的计算环境不使用 AWSServiceRoleForBatch 服务相关角色时
- 当您需要在不同的服务角色类型之间转换时

何时建议使用蓝绿更新

零停机时间对工作负载至关重要的生产环境尤其建议使用蓝绿更新。这种方法在您需要在转移生产工作负载之前测试新配置，确保更改满足您的性能和可靠性要求时非常有效。当快速回滚功能对运营十分重要时，尤其是在更新具有重大更改的自定义 AMI 时，可选择蓝绿更新。当您想在完全提交更改之前验证性能特征和行为，确保更新过程顺畅无误时，也非常适合使用此方法。

先决条件

在执行蓝绿更新之前，请确保您已满足下列要求：

- 具有创建和管理计算环境所需的适当 IAM 权限
- 具有查看和修改作业队列设置的访问权限
- 为您的作业定义配置了作业重试策略，用于处理转移期间可能出现的作业失败。有关更多信息，请参阅 自动作业重试。

- 拥有新计算环境的 AMI ID。您可以指定：
  - 最近批准的 Amazon ECS 优化型 AMI 版本（默认使用）
  - 符合 Amazon ECS 容器实例 AMI 规范的自定义 AMI。您可以通过以下方式之一来指定自定义 AMI：
    - 使用 EC2 配置中的映像 ID 覆盖字段
    - 在启动模板中指定

    有关创建自定义 AMI 的更多信息，请参阅教程：创建计算资源 AMI。

在创建新环境之前，您需要记录现有计算环境的配置。您可以使用 AWS 管理控制台或 AWS CLI 完成此操作。

> **Note**
>
> 以下过程详细说明了如何执行仅更改 AMI 的蓝绿更新。您可以更新新环境的其他设置。

> **Important**
>
> 当您移除旧（蓝色）计算环境时，所有当前正在这些实例上运行的作业都将失败，因为这些实例将被终止。在作业定义中配置作业重试策略以自动处理这些失败。有关更多信息，请参阅 自动作业重试。
> 确信新环境能够正常运行后：
>
> 1. 编辑作业队列以移除旧计算环境。
> 2. 等待旧环境中所有仍在运行的作业完成运行。
> 3. 删除旧的计算环境。

Performing blue/green updates using the AWS 管理控制台

1. 克隆当前计算环境
   a. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
   b. 选择您现有的计算环境。
   c. 选择操作，然后选择克隆。

d. 对于名称，输入新计算环境的唯一名称。

e. 选择下一步。

f. 在实例配置部分中，更新 AMI 设置：

i. 展开其他配置。

ii. 对于 EC2 配置，请在映像类型中指定新的 AMI 类型，并在映像 ID 覆盖字段中指定 AMI ID。

g. 选择下一步。

h. 对于网络配置，请选择下一步。

i. 检查从现有环境中自动复制的其他设置。

j. 选择创建计算环境。

k. 等待新计算环境的状态变为 VALID。

2. 更改作业队列顺序

a. 在导航窗格中，选择 作业队列。

b. 选择与您的现有计算环境关联的作业队列。

c. 选择编辑。

d. 在已连接的计算环境下，添加新的计算环境：

- 为新计算环境添加一个比现有环境更高的顺序号以转移工作负载。
- 验证新环境能够正常运行后，可以通过为新环境提供一个更低的顺序号，从而使其成为主环境。

e. 选择更新作业队列。

3. 清理

a. 监控新环境中的作业执行情况，确保一切按预期运行。

b. 确信新环境能够正常运行后：

1. 编辑作业队列以移除旧计算环境。

2. 等待旧环境中所有仍在运行的作业完成运行。

3. 删除旧的计算环境。

Performing blue/green updates using the AWS CLI

1. 要使用 AWS CLI 获取配置，请使用以下命令：

```
aws batch describe-compute-environments \
  --compute-environments your-compute-environment-name
```

保存输出以备在创建新环境时参考。

2. 使用现有环境中的配置创建新计算环境，不过要使用新的 AMI。示例命令结构如下：

将示例值替换为上一步中的实际配置：

```
cat <<EOF > ./blue-green-compute-environment.json
{
  "computeEnvironmentName": "your-new-compute-environment-name",
  "type": "MANAGED",
  "state": "ENABLED",
  "computeResources": {
    "instanceRole": "arn:aws:iam::012345678901:instance-profile/
ecsInstanceRole",
    "type": "EC2",
    "minvCpus": 2,
    "desiredvCpus": 2,
    "maxvCpus": 256,
    "instanceTypes": [
      "optimal"
    ],
    "allocationStrategy": "BEST_FIT_PROGRESSIVE",
    "ec2Configuration": [
      {
        "imageType": "ECS_AL2023",
        "imageIdOverride": "ami-0abcdef1234567890"
      }
    ],
    "subnets": [,
      "subnet-0abcdef1234567890"
    ],
    "securityGroupIds": [
      "sg-0abcdef1234567890"
    ]
  }
}
```

```
EOF
```

```
$ aws batch create-compute-environment --cli-input-json file://./blue-green-compute-environment.json
```

3. 等待新环境的状态变为可用：

```
aws batch describe-compute-environments \
  --compute-environments your-new-compute-environment-name \
  --query 'computeEnvironments[].status'
```

4. 将新计算环境添加到您的作业队列：

```
aws batch update-job-queue \
  --job-queue your-job-queue \
  --compute-environment-order order=1,computeEnvironment=your-existing-environment \
  order=2,computeEnvironment=your-new-compute-environment-name
```

5. 完成验证后，再次更新以使新环境成为主环境：

```
aws batch update-job-queue \
  --job-queue your-job-queue \
  --compute-environment-order order=1,computeEnvironment=your-new-compute-environment-name
```

旧环境中的所有作业完成后禁用旧环境，然后将其删除：

```
aws batch update-compute-environment \
    --compute-environment your-existing-environment \
    --state DISABLED
```

```
aws batch delete-compute-environment \
  --compute-environment your-existing-environment
```

计算资源 &AMI;

默认情况下，AWS Batch托管计算环境使用最近批准的 Amazon ECS 优化 AMI 版本作为计算资源。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 不会升级该计算环境中的 AMI。当有更新版本的 Amazon ECS 优化的 AMI 可用时，AWS Batch 也不会更新该计算环境中的 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。要为 AWS Batch 作业使用新的 AMI，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 在 AWS Batch 中更新计算环境。要使用计算环境的增强更新来更新 AMI，请遵循以下规则：
>
> - 不设置服务角色（`serviceRole`）参数，或将其设置为 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，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 规范包含：

（必需）

- 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 为 GitHub 上的 Amazon ECS 优化 AMI 的 Linux 变体提供了更改日志。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 `ecs-init` 的更多信息，请参阅 GitHub 上的 `ecs-init` 项目。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 代理未在计算资源上运行，则无法接受来自 AWS Batch 的任务。

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。

教程：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 必须与您打算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 Amazon ECS 优化型 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《亚马逊弹性容器服务开发人员指南》中的在 Amazon Linux 2 EC2 实例上安装 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，请参阅亚马逊 EC2 用户指南中的启动 EC2 实例和亚马逊实例存储。
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。为此，请在创建 AWS Batch 计算环境时选择映像类型并在映像 ID 覆盖框中输入自定义 AMI ID。有关更多信息，请参阅 the section called “教程：使用 Amazon EC2 资源创建托管计算环境”。

   > **Note**
   >
   > 您为计算环境选择的 AMI 必须与您打算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 Amazon ECS 优化型 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。

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 教程：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。

在托管式计算环境中，如果计算环境指定任何 p6, p3, p4, p5, g3, g3s, g4、g5 或 g6 实例类型或实例系列，则 AWS Batch 会使用 Amazon ECS GPU 优化型 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，并使用最新的版本的默认 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

Amazon Linux AMI（也称为 Amazon Linux 1）已于 2023 年 12 月 31 日达到其终止使用时间。AWS Batch 已终止对 Amazon Linux AMI 的支持，因为它将自 2024 年 1 月 1 日起停止接收任何安全更新或错误修复。有关 Amazon Linux 生命周期终止的更多信息，请参阅 AL 常见问题。

我们建议您将现有的基于 Amazon Linux 的计算环境更新到 Amazon Linux 2023，以防止不可预见的工作负载中断并继续接收安全更新和其他更新。

您使用 Amazon Linux AMI 的计算环境可能会在 2023 年 12 月 31 日生命周期终止日期之后继续运行。但是，这些计算环境将不再接收来自 AWS 的任何新软件更新、安全补丁或错误修复。在使用寿命终止后，您有责任在 Amazon Linux AMI 上维护这些计算环境。我们建议将 AWS Batch 计算环境迁移到 Amazon Linux 2023 或 Amazon Linux 2，以保持最佳性能和安全性。

要获得将 AWS Batch 从 Amazon Linux AMI 迁移到 Amazon Linux 2023 或 Amazon Linux 2 方面的帮助，请参阅 Updating compute environments - AWS Batch。

Amazon EKS Amazon Linux 2 AMI 弃用

AWS 将于 2025 年 11 月 26 日起终止对 Amazon EKS 优化型 Amazon Linux 2 AMI 的支持。我们建议在 2025 年 11 月 26 日前将 AWS Batch Amazon EKS 计算环境迁移到 Amazon Linux 2023，以确保最佳性能和安全性。

尽管在 2025 年 11 月 26 日终止支持之日后，您可以继续在 Amazon EKS 计算环境中使用 Batch 提供的 Amazon EKS 优化型 Amazon Linux 2 AMI，但这些计算环境将不再能够获得来自 AWS 的任何新软件更新、安全补丁或错误修复。在生命周期终止后，您自行负责维护 Amazon EKS 优化型 Amazon Linux AMI 上的这些计算环境。

有关 Amazon EKS AL2 生命周期终止的更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon EKS AMI 弃用常见问题。

有关将 AWS Batch Amazon EKS 计算环境从 Amazon Linux 2 迁移到 Amazon Linux 2023 的帮助，请参阅如何从 EKS 升级 AL2 到 EKS AL2023。

Amazon ECS Amazon Linux 2 AMI 弃用

AWS 将终止对 Amazon Linux 2 的支持。从 2026 年 1 月起，AWS Batch 将新 Amazon ECS 计算环境的默认 AMI 从 Amazon Linux 2 更改为 Amazon Linux 2023。我们建议将 AWS Batch Amazon ECS 计算环境迁移到 Amazon Linux 2023，以保持最佳性能和安全性。

有关 Amazon Linux 2 生命周期终止的更多信息，请参阅 Amazon Linux 2 FAQs。

有关 Amazon Linux 2 和 Amazon Linux 2023 之间差异的信息，请参阅《Amazon Linux 2023 用户指南》中的 Compare Amazon Linux 2023 and Amazon Linux 2。

有关适用于 Amazon ECS 优化型 AMI 的 Amazon Linux 2023 更改的信息，请参阅《Amazon ECS 用户指南》中的从 Amazon Linux 2 迁移到 Amazon Linux 2023 Amazon ECS 优化版 AMI。

有关将 AWS Batch Amazon ECS 计算环境从 Amazon Linux 2 迁移到 Amazon Linux 2023 的帮助，请参阅如何从 ECS AL2 迁移到 ECS AL2023。

使用 Amazon EC2 启动模板和 AWS Batch

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，您可以修改 AWS Batch 计算资源的默认配置，而无需创建自定义配置 AMIs。

> **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 IDs
- 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 在 AWS Batch 中更新计算环境。

默认启动模板和覆盖启动模板

您可以为计算环境定义默认启动模板，并为特定实例类型和系列定义覆盖启动模板。这可能会非常实用，因此计算环境中的大多数实例类型都使用默认模板。

可以使用替换变量 `$Default` 和 `$Latest`，而不指定特定的版本。如果未提供覆盖启动模板，则会自动应用默认启动模板。

如果您使用`$Default`或`$Latest`变量，则 AWS Batch 将在创建计算环境时应用当前信息。如果 future 的默认版本或最新版本发生变化，则必须通过UpdateComputeEnvironment或通过 AWS 管理控制台 -来更新信息 AWS Batch。

为提供更好的灵活性，您可以定义将应用于特定计算实例类型或系列的覆盖启动模板。

> **Note**
>
> 您最多可以为每个计算环境指定十（10）个覆盖启动模板。

使用 `targetInstanceTypes` 参数来选择应使用此覆盖启动模板的实例类型或系列。实例类型或系列必须首先由 `instanceTypes` 参数标识。

如果您定义了覆盖启动模板，后来决定将其移除，则可以在 `UpdateComputeEnvironment` API 操作中传递一个空数组来取消设置 `overrides` 参数。您也可以选择在提交 `UpdateComputeEnvironment` API 操作时不包含 `overrides` 参数。欲了解更多信息，请参阅 `LaunchTemplateSpecification.overrides`

有关更多信息，请参阅 AWS Batch API 参考指南`LaunchTemplateSpecificationOverride.targetInstanceTypes`中的。

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。

> **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，然后在调用时将该字符串作为UserData参数值提交 CreateLaunchTemplate，如此 JSON 文件所示。
>
> ```
> {
>     "LaunchTemplateName": "base64-user-data",
>     "LaunchTemplateData": {
>         "UserData":
>  "ewogICAgIkxhdW5jaFRlbXBsYXRlTmFtZSI6ICJpbmNyZWFzZS1jb250YWluZXItdm9sdW..."
>     }
> ```

```
}
```

主题

- 参考：Amazon EC2 启动模板示例

参考：Amazon EC2 启动模板示例

以下是 MIME 分段文件的示例，您可以用它来创建您自己的模板。

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：挂载现有的 Amazon f FSx or Lustre 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：挂载现有的 Amazon f FSx or Lustre 文件系统

Example

此示例 MIME 多部分文件将计算资源配置为从 Extras Library 安装`lustre2.10`软件包，并将现有 FSx 的 For Lustre 文件系统挂载到`/scratch`且挂载名称为。`fsx`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 Amazon FSx for Lustre 用户指南中的安装 Lustre 客户端。有关更多信息，请参阅 Amazon FSx for Lustre 用户指南中的自动挂载您的亚马逊 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"
        }
    ],
}
```

实例元数据服务（IMDS）配置

实例元数据服务（IMDS）向在 EC2 实例上运行的应用程序提供有关该等实例的元数据。使用 IMDSv2 处理所有新工作负载，并将现有工作负载从 IMDSv1 迁移到 IMDSv2 以提高安全性。有关 IMDS 和配置 IMDS 的更多信息，请参阅《Amazon EC2 用户指南》中的使用实例元数据管理 EC2 实例和为新实例配置实例元数据选项。

配置方案

根据计算环境设置选择适当的配置方法：

没有启动模板的默认 AMI

当您使用默认 AWS Batch AMI 但未指定启动模板时，请选择以下选项之一：

1. 使用Amazon Linux 2023 的默认 AMI：Amazon Linux 2023 默认要求使用 IMDSv2。创建计算环境时，对于映像类型请选择 Amazon Linux 2023。

2. 设置账户级别 IMDSv2 配置：将您的 AWS 账户配置为要求所有新实例使用 IMDSv2。此设置会影响您在该账户中启动的所有新实例。有关说明，请参阅《Amazon EC2 用户指南》中的将 IMDSv2 设置为账户默认设置。

> **Note**
>
> 启动模板或 AMI 配置可能会覆盖账户级别 IMDS 配置。启动模板设置的优先级高于账户级别设置。

没有启动模板的自定义 AMI

当您在没有启动模板的情况下使用自定义 AMI 时，请选择以下选项之一：

1. 以 Amazon Linux 2023 为基础：将 Amazon Linux 2023 作为基础映像来构建自定义 AMI。有关创建适用于 Batch 的自定义 AMI 的更多信息，请参阅教程：创建计算资源 AMI。
2. 在自定义 AMI 中配置 IMDSv2：创建自定义 AMI 时，将其配置为要求使用 IMDSv2。有关操作说明，请参阅《Amazon EC2 用户指南》中的为自定义实例配置实例元数据选项。
3. 设置账户级别 IMDSv2 配置：将您的 AWS 账户配置为要求所有新实例使用 IMDSv2。此设置会影响您在该账户中启动的所有新实例。有关说明，请参阅《Amazon EC2 用户指南》中的将 IMDSv2 设置为账户默认设置。

> **Note**
>
> 启动模板或 AMI 配置可能会覆盖账户级别 IMDS 配置。启动模板设置的优先级高于账户级别设置。

使用启动模板

在计算环境中使用启动模板时，请在启动模板中添加元数据选项以要求使用 IMDSv2。有关将启动模板与 Batch 结合使用的更多信息，请参阅使用 Amazon EC2 启动模板和 AWS Batch。

```
{
    "LaunchTemplateName": "batch-imdsv2-template",
    "VersionDescription": "IMDSv2 only template for Batch",
    "LaunchTemplateData": {
        "MetadataOptions": {
            "HttpTokens": "required"
```

```
        }
    }
}
```

使用 AWS CLI 创建启动模板：

```
aws ec2 create-launch-template --cli-input-json file://imds-template.json
```

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)。从 2026 年 1 月起，默认值将更改为 AL2023 (ECS_AL2023)。

AWS 将终止对 Amazon Linux 2 的支持。我们建议将 AWS Batch Amazon ECS 计算环境迁移到 Amazon Linux 2023，以保持最佳性能和安全性。有关更多信息，请参阅 Amazon ECS Amazon Linux 2 AMI 弃用。

我们建议您将现有的基于 Amazon Linux 的计算环境更新到 Amazon Linux 2023，以防止不可预见的工作负载中断并继续接收安全更新和其他更新。

要获得有关将 AWS Batch 从 Amazon Linux AMI 迁移到 Amazon Linux 2023 的帮助，请参阅如何从 ECS AL2 迁移到 ECS AL2023

主题

- 如何从 ECS AL2 迁移到 ECS AL2023

如何从 ECS AL2 迁移到 ECS AL2023

AL2023 是一款基于 Linux 的操作系统，旨在为云应用程序提供安全、稳定和高性能的环境。有关 AL2 和 AL2023 之间差异的详细信息，请参阅《Amazon Linux 2023 用户指南》中的 Compare Amazon Linux 2023 and Amazon Linux 2。

从 2026 年 1 月起，AWS Batch 将新 Amazon ECS 计算环境的默认 AMI 从 Amazon Linux 2 更改为 Amazon Linux 2023，因为 AWS 将终止对 Amazon Linux 2 的支持。如果您在创建新计算环境时没有为 imageType.Ec2Configuration 字段指定值，则将使用默认 AMI。我们建议将 AWS Batch Amazon ECS 计算环境迁移到 Amazon Linux 2023，以保持最佳性能和安全性。

您可以使用以下路径之一从 AL2 升级到 AL2023，具体取决于您的计算环境配置。

使用 Ec2Configuration.ImageType 升级

- 如果您没有使用启动模板或启动模板覆盖功能，请将 Ec2Configuration.ImageType 更改为 ECS_AL2023（如果您使用的是 GPU 实例，则请更改为 ECS_AL2023_NVIDIA），然后运行 UpdateComputeEnvironment。
- 如果您指定了 Ec2Configuration.ImageIdOverride，Ec2Configuration.ImageType 必须与 Ec2Configuration.ImageIdOverride 中指定的 AMI 类型匹配。

  如果 ImageIdOverride 和 ImageType 不匹配，则计算环境可能无法正常运行。

使用启动模板升级

- 如果您使用启动模板来指定基于 ECS_AL2023 的 AMI，请确保您的启动模板与 Amazon Linux 2023 兼容。有关适用于 Amazon ECS 优化型 AMI 的 Amazon Linux 2023 更改的信息，请参阅《Amazon ECS 用户指南》中的从 Amazon Linux 2 迁移到 Amazon Linux 2023 Amazon ECS 优化版 AMI。
- 对于 AL2023 AMI，请验证所有自定义用户数据或初始化脚本是否与 AL2023 环境和软件包管理系统兼容。

使用 CloudFormation 升级

- 如果您使用 CloudFormation 管理计算环境，请更新模板以将 Ec2Configuration 中的 ImageType 属性从 ECS_AL2 更改为 ECS_AL2023（如果您使用的是 GPU 实例，则更改为 ECS_AL2023_NVIDIA）：

  ```
  ComputeEnvironment:
    Type: AWS::Batch::ComputeEnvironment
    Properties:
      ComputeResources:
        Ec2Configuration:
          - ImageType: ECS_AL2023
  ```

  然后更新 CloudFormation 堆栈以应用更改。
- 如果您的 CloudFormation 模板使用 ImageIdOverride 指定了某个自定义 AMI，请确保 AMI ID 对应于某个基于 AL2023 的 AMI 并且与 ImageType 设置一致。

迁移注意事项

从 Amazon Linux 2 迁移到 Amazon Linux 2023 时，应注意以下因素：

- 软件包管理：Amazon Linux 2023 使用 dnf 而不是 yum 来进行软件包管理。
- 系统服务：AL2 和 AL2023 的某些系统服务及其配置可能不同。
- 容器运行时：AL2 和 AL2023 都支持 Docker，但是 AL2023 的默认配置可能不同。
- 安全性：AL2023 包括增强的安全功能，可能需要更新安全性相关配置。
- 实例元数据服务版本 2（IMDSv2）：IMDSv2 是一项面向会话的服务，需要基于令牌的身份验证才能访问 EC2 实例元数据，从而增强了安全性。有关 IMDS 的更多信息，请参阅《Amazon EC2 用户指南》中的实例元数据服务版本 2 的工作原理。

有关更改和迁移注意事项的完整列表，请参阅《Amazon ECS 用户指南》中的从 Amazon Linux 2 迁移到 Amazon Linux 2023 Amazon ECS 优化版 AMI。

AWS Batch 的实例类型分配策略

创建托管计算环境后，AWS Batch将从指定的`instanceTypes`中选择最适合作业要求的实例类型。分配策略定义当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会选择最适合作业要求的实例类型，并优先考虑成本最低的实例类型。如果选定实例类型没有额外实例可用，AWS Batch将等待额外实例可用。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 在 AWS Batch 中更新计算环境。

> **Note**
>
> AWS Batch 管理您账户中的 AWS 资源。默认情况下，具有 BEST_FIT 分配策略的计算环境最初使用了启动配置。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 计算环境将默认为启动模板。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会选择足够大的额外实例类型，以满足队列中作业的要求。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。

> **Note**
>
> 对于多节点并行作业，AWS Batch 会选择可用的最佳实例类型。如果该实例类型因容量不足而不可用，则不会启动该系列中的其他实例类型。

`SPOT_CAPACITY_OPTIMIZED`

AWS Batch会选择一个或以上足够大的实例类型，以满足队列中作业的要求。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` 策略使用按需型实例或竞价型实例，`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`maxvCpus`以满足容量要求。在这种情况下，AWS Batch始终不会超过`maxvCpus`一个以上的实例。

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。

主题

- 预留系统内存
- 教程：查看计算资源内存
- Amazon EKS 上 AWS Batch 的内存和 vCPU 注意事项

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。我们建议为 Amazon ECS 容器代理和其他关键系统进程预留 5% 的内存缓冲区。

教程：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。

   总容量内存值是计算资源首次启动时注册到 Amazon ECS 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。

示例：作业定义

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。

示例：节点 CPU 预留

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。

示例：节点内存预留

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。

Fargate 计算环境

Fargate 是一项无需管理 Amazon EC2 实例的服务器或集群即可运行容器的技术。 AWS Batch 使用 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。

主题

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：

- 大于 16 v CPUs
- 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 Fargate 服务配额。

Fargate 上的作业定义

AWS Batch job AWS Fargate s on 不支持所有可用的作业定义参数。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，如何解决 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，请参阅 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，请参阅 AWS Batch IAM 执行角色。

Fargate 上的作业队列

AWS Batch 开启的任务队 AWS Fargate 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

AWS Batch 开启的计算环境 AWS Fargate 不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。

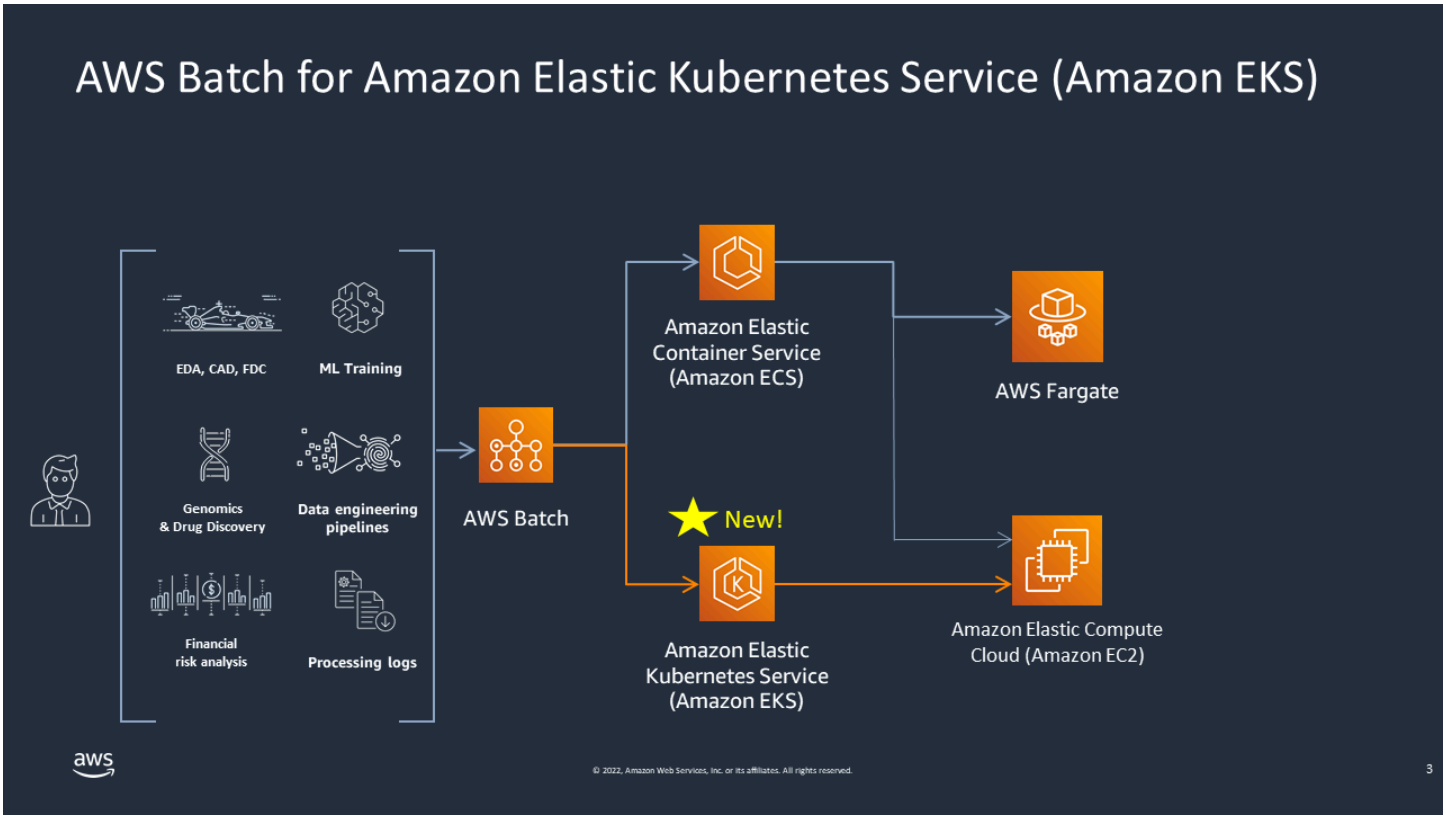
`type`

必须是 `FARGATE` 或 `FARGATE_SPOT`。

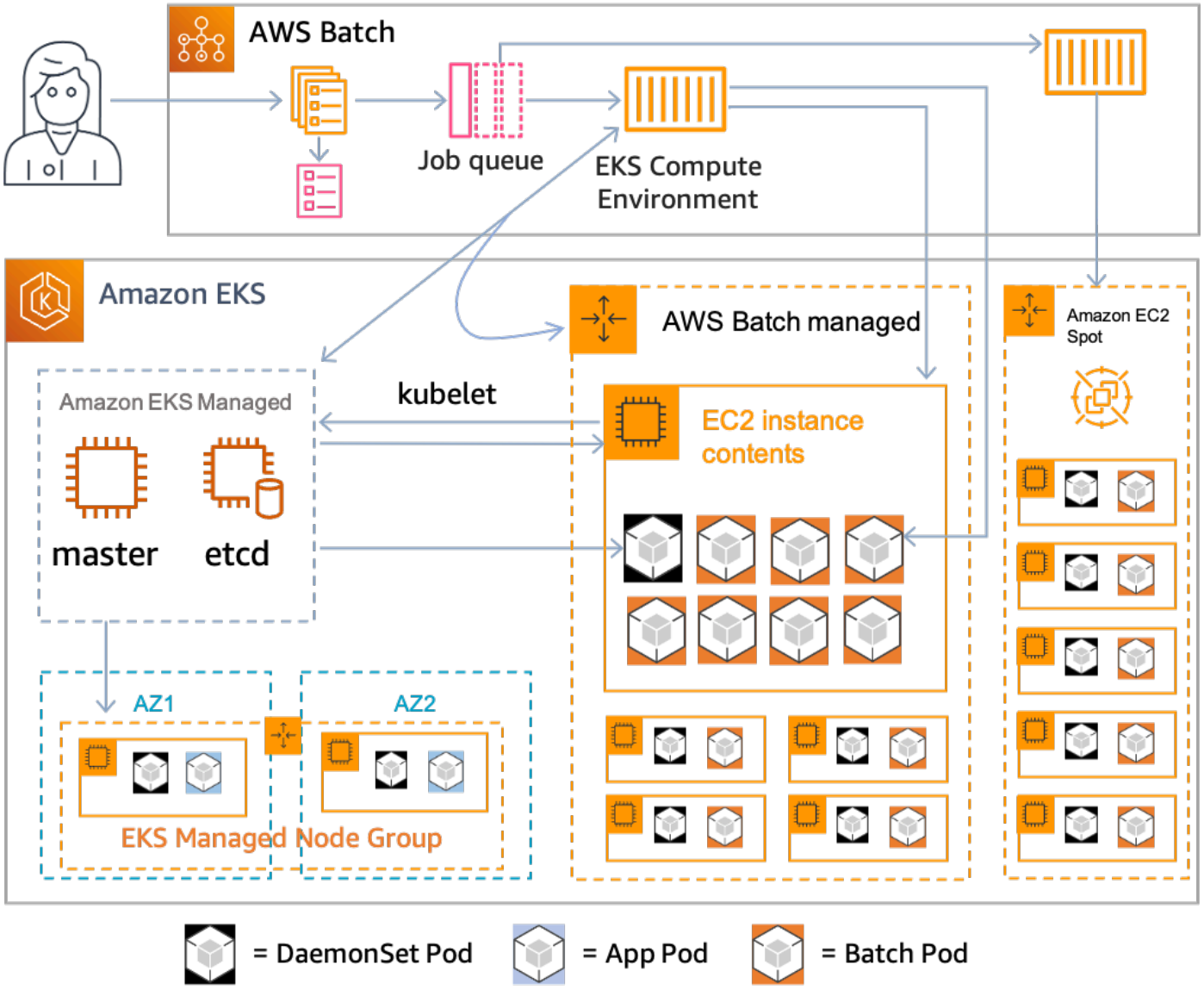
```
"type": "FARGATE_SPOT"
```

Amazon EKS 计算环境

开始使用 Amazon EKS 上的 AWS Batch提供了创建 EKS 计算环境的简短指南。本节提供了有关 Amazon EKS 计算环境的更多详细信息。



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Kubernetes APIs 用创建、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 开始使用 Amazon EKS 上的 AWS Batch。

Amazon EKS

主题

- Amazon EKS 默认 AMI
- 混合 AMI 环境
- 支持的Kubernetes版本
- 更新计算环境的 Kubernetes 版本
- Kubernetes节点的共同责任
- DaemonSet在 AWS Batch 托管节点上运行
- 自定义 Amazon EKS 启动模板
- 如何从 EKS 升级 AL2 到 EKS AL2023

Amazon EKS 默认 AMI

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经过优化 Amazon EKS 的 AMI。一般情况下，我们建议您使用默认 AMI 选择。有关优化的亚马逊 EKS 的更多信息 AMIs，请参阅亚马逊 EKS 用户指南 AMIs中的亚马逊 EKS 优化亚马逊 Linux。

> **Important**
>
> 亚马逊 Linux 2023 AMIs 是亚马逊 EKS AWS Batch 的默认开启。
> AWS 将从 25 年 11 月 26 日起终止 AL2对 Amazon EKS 的支持 AL2（经过优化和加速 AMIs）。在 25 年 11 月 26 AMIs end-of-support 日之后，您可以继续在您的 Amazon EKS 计

算环境中使用 AWS Batch经过优化 Amazon EKS 的 Amazon Linux 2，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有关从升级 AL2 到的更多信息 AL2023，请参阅如何从 EKS 升级 AL2 到 EKS AL2023《AWS Batch 用户指南》。

运行以下命令查看为您的 Amazon EKS 计算环境 AWS Batch 选择了哪种 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 1.32 AMI,
 depending on instance types
    $ aws batch describe-compute-environments --compute-environments My-Eks-CE1 \
        | jq '.computeEnvironments[].computeResources.ec2Configuration'
    [
      {
        "imageType": "EKS_AL2",
        "imageKubernetesVersion": "1.32"
      }
    ]
```

以下示例是 GPU 实例类型。

```
# GPU CE example: indicates Batch has choosen the AL2 x86 EKS Accelerated 1.32 AMI
    $ aws batch describe-compute-environments --compute-environments My-Eks-GPU-CE \
        | jq '.computeEnvironments[].computeResources.ec2Configuration'
    [
      {
        "imageType": "EKS_AL2_NVIDIA",
        "imageKubernetesVersion": "1.32"
      }
    ]
```

混合 AMI 环境

您可以使用启动模板替换来创建同时使用亚马逊 Linux 2 (AL2) 和亚马逊 Linux 2023 (AL2023) AMIs 的计算环境。这对于为不同的架构使用不同的 AMIs 架构或在从过渡 AL2 到 AL2023的迁移期间使用不同的方法非常有用。

> **Note**
>
> AWS 将从 25 年 11 月 26 日起终止 AL2对 Amazon EKS 的支持 AL2（经过优化和加速 AMIs）。虽然您可以在 25 年 11 月 26 AMIs end-of-support 日之后继续在您的 Amazon EKS

计算环境中使用 AWS Batch经过优化 Amazon EKS 的 Amazon Linux 2，但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在过渡期间，混合 AMI 环境非常有用，允许您逐步将工作负载迁移到， AL2023 同时保持与现有工作负载 AL2的兼容性。

同时使用这两种 AMI 类型的配置示例：

```
{
  "computeResources": {
    "launchTemplate": {
      "launchTemplateId": "TemplateId",
      "version": "1",
      "userdataType": "EKS_BOOTSTRAP_SH",
      "overrides": [
        {
          "instanceType": "c5.large",
          "imageId": "ami-al2-custom",
          "userdataType": "EKS_BOOTSTRAP_SH"
        },
        {
          "instanceType": "c6a.large",
          "imageId": "ami-al2023-custom",
          "userdataType": "EKS_NODEADM"
        }
      ]
    },
    "instanceTypes": ["c5.large", "c6a.large"]
  }
}
```

支持的Kubernetes版本

AWS Batch 在 Amazon 上，EKS 目前支持以下Kubernetes版本：

- 1.34
- 1.33
- 1.32
- 1.31
- 1.30
- 1.29

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。您可以在更新 Amazon EKS 集群控制面板版本之前或之后，在其 Amazon EKS 节点上执行 Kubernetes 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=1.32}]'
```

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。

- 不要将你的 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 AMIs。您要负责更新它们。运行以下命令将您的 AMI 更新 AMIs 到最新 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。运行以下命令将您的计算机环境Kubernetes版本更新为*1.32*。

```
$ aws batch update-compute-environment \
    --compute-environment <compute-environment-name> \
    --compute-resources \
      'ec2Configuration=[{imageType=EKS_AL2,imageKubernetesVersion=1.32}]'
```

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，请参阅在 AWS Batch 中更新计算环境和 UpdateComputeEnvironment API 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"
```

自定义 Amazon EKS 启动模板

AWS Batch 在 Amazon 上，EKS 支持启动模板。启动模板的功能受到限制。

Important

- 对于 EKS 来说 AL2 AMIs， AWS Batch 跑步`/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。
- 对于 EKS AL2023， AWS Batch 利用来NodeConfigSpec自 EKS 的使实例加入 EKS 集群。AWS Batch 为 EK NodeConfigSpecS 集群填充 ClusterDetails，您无需指定它们。

Note

我们建议您不要在启动模板中NodeConfigSpec设置以下任何设置，因为 AWS Batch 这将覆盖您的值。有关更多信息，请参阅 Kubernetes节点的共同责任。

- `Taints`
- `Cluster Name`
- `apiServerEndpoint`
- `certificatAuthority`
- `CIDR`
- 不要创建使用前缀 `batch.amazonaws.com/` 的标签

> **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

AWS Batch 支持在`kubelet`命令中添加额外的参数。有关支持的参数列表，请参阅Kubernetes文档中的kubelet。在以下 EKS 示例中 AL2 AMIs*`--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==--
```

对于 EKS AL2023 AMIs ，文件格式为 YAML。有关支持的参数列表，请参阅Kubernetes文档中的NodeConfigSpec。在以下 EKS 示例中 AL2023 AMIs*`--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: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  kubelet:
    flags:
    - --node-labels=mylabel=helloworld

--==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。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。

如果您使用的是优化型 EKS_AL2023 或 EKS_AL2023_NVIDIA AMI，则无需指定容器运行时，因为仅支持 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==--
```

挂载 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，卷装载在 /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，我们建议您查看 Amazon EKS 用户指南中为 pod 和服务分配 IPv6 地址中概述的注意事项和条件。

如何从 EKS 升级 AL2 到 EKS AL2023

经过优化的亚马逊 EKS AMIs 有两个系列可供选择，分别基于亚马逊 Linux 2 (AL2) 和亚马逊 Linux 2023 (AL2023)。 AL2023 是基于 Linux 的操作系统，旨在为您的云应用程序提供安全、稳定和高性能的环境。有关两者之间 AL2 差异的更多信息， AL2023 请参阅亚马逊 E KS 用户指南中的从亚马逊 Linux 2 升级到亚马逊 Linux 2023。

> **Important**
>
> AWS 将从 25 年 11 月 26 日起终止 AL2对 Amazon EKS 的支持 AL2（经过优化和加速 AMIs）。我们建议在 25 年 11 月 26 日之前将 AWS Batch Amazon EKS 计算环境迁移到 Amazon Linux 2023，以保持最佳性能和安全性。虽然您可以在 25 年 11 月 26 AMIs end-of-

support 日之后继续在您的 Amazon EKS 计算环境中使用 AWS Batch经过优化 Amazon EKS 的 Amazon Linux 2，但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，你有责任在亚马逊 EKS 优化的 Amazon Linux 2 AMI 上维护这些计算环境 end-of-life。

根据计算环境的配置方式，您可以使用以下从 AL2 到的升级路径之一 AL2023。

使用 Ec2 配置进行升级。 ImageType

- 如果您没有使用启动模板或启动模板替代，请更改 Ec2Configuration。 ImageType到EKS_AL2023或EKS_AL2023_NVIDIA然后运行UpdateComputeEnvironment。
- 如果您指定 Ec2 配置。 ImageIdOverride然后是 Ec2 配置。 ImageType必须与 Ec 2Configuration 中指定的 AMI 类型相匹配。 ImageIdOverride。

  如果 ImageIdOverride 与 ImageType 不匹配，则该节点将无法加入集群。

使用启动模板升级

- 如果您在启动模板或启动模板覆盖中定义了任何kubelet额外的参数，则需要将它们更新为新的kubelet额外参数格式。

  如果与 kubelet 附加参数格式不匹配，则不会应用附加参数。
- 因为 AL2023 AMIs，c ontainerd 是唯一支持的容器运行时。无需在启动模板中指定 EKS_AL2023 到容器运行时。

  您不能使用指定自定义容器运行时EKS_AL2023。
- 如果使用指定了基于 EKS_AL2023 的 AMI 的启动模板或启动模板覆盖，则需要将 userdataType 设置为 EKS_NODEADM。

  如果 userdataType 与 AMI 不匹配，则该节点将无法加入 EKS 集群。

的服务环境 AWS Batch

服务环境可以 AWS Batch 与 SageMaker AI 集成。服务环境包含提交和管理 SageMaker训练作业所需 AWS Batch 的 SageMaker AI 特定配置参数，同时提供 AWS Batch队列、计划和优先级管理功能。

在服务环境中，数据科学家和机器学习工程师可以向服务作业队列提交优先级的 SageMaker 培训作业。通过这种集成，您将不再需要手动协调机器学习工作负载，防止意外超支，并提高组织机器学习工作流的资源利用率。

主题

什么是 AWS Batch中的服务环境

服务环境是一种 AWS Batch 资源，其中包含 AWS Batch 与 SageMaker AI 集成所需的配置参数。服务环境 AWS Batch 允许提交和管理 SageMaker 培训作业，同时提供 AWS Batch队列、计划和优先级管理功能。

服务环境可解决数据科学团队在管理机器学习工作负载时面临的常见挑战。组织经常会限制可用于训练模型的实例数量，以防止意外超支、满足预算约束、使用预留实例节省成本，或者为工作负载使用特定的实例类型。但数据科学家需要同时运行的工作负载数可能会超过所分配实例的容量，因此需要手动协调才能决定何时运行哪些工作负载。

这种协调方面的挑战会影响各类规模的组织，无论是仅有少数数据科学家的小型团队，还是大型规模化运营的企业。随着组织的发展，复杂性也随之增加，需要更多时间来管理工作负载的协调，而且经常会需要基础设施管理员的参与。这些手动操作会浪费时间并降低实例效率，进而给客户带来实际成本损失。

在服务环境中，数据科学家和机器学习工程师可以将具有优先级的 SageMaker 培训作业提交到可配置队列，从而确保工作负载在资源可用后立即自动运行，无需干预。这种集成利用 AWS Batch了广泛的排队和日程安排功能，使客户能够自定义其排队和日程安排策略以符合其组织的目标。

服务环境如何与其他 AWS Batch 组件配合使用

服务环境与其他 AWS Batch 组件集成以实现 SageMaker 训练作业队列：

- Job qu eues-服务环境与作业队列相关联，使队列能够处理 SageMaker 训练作业的服务作业
- 服务作业-当您向与服务环境关联的队列提交服务作业时， AWS Batch 使用该环境的配置提交相应的 SageMaker 训练作业
- 调度策略-服务环境使用 AWS Batch 调度策略来确定 SageMaker 训练作业的优先级并管理其执行顺序

这种集成使您可以利用成熟 AWS Batch的排队和调度功能，同时保持 SageMaker 训练作业的全部功能和灵活性。

有关服务环境的最佳实践

服务环境提供了大规模管理 SageMaker 培训作业的功能。遵循以下最佳实践有助您优化成本、性能和运营效率，同时避免可能影响机器学习工作流的常见配置问题。

在规划服务环境容量时，请考虑适用于 SageMaker 训练作业队列的特定配额和限制。每个服务环境都有以实例数表示的最大容量限制，它直接控制可以同时运行多少 SageMaker 训练作业。了解这些限制有助于防止资源争用，并确保作业执行时间的可预测性。

最佳的服务环境性能取决于对 SageMaker 训练作业调度的独特特征的理解。与传统的容器化作业不同，服务作业通过SCHEDULED状态过渡，而 SageMaker AI 会获取和配置必要的训练实例。这意味着作业启动时间可能会因实例可用性和区域容量而有显著差异。

> **Important**
>
> 服务环境具有特定的配额，这可能会影响您扩展 SageMaker 培训工作负载的能力。每个账户最多可以创建 50 个服务环境，并且每个作业队列仅支持一个关联的服务环境。此外，单个作业的服务请求有效载荷限制为 10 KiB，SubmitServiceJob API 限制为每个账户每秒 5 个事务处理。在容量计划期间了解这些限制有助于防止意外的扩缩约束。

要有效监控服务环境，需要同时 AWS Batch 关注 SageMaker 人工智能服务指标。通过作业状态变换可以深入了解系统性能，尤其是处于 SCHEDULED 状态的时间，因为这可以指示容量可用性模式，因此十分宝贵。与计算环境类似，服务环境维护自己的生命周期状态，即会经过 CREATING、VALID、INVALID 和 DELETING 状态变换，并且应对这些状态进行监控以确定运行状

况。具有成熟监控实践的组织通常会跟踪队列深度、作业完成率和实例利用率模式，以不断优化其服务环境配置。

中的服务环境状态和生命周期 AWS Batch

服务环境维护生命周期状态，这些状态表明其当前的运行状态和处理 SageMaker 培训作业的准备情况。了解这些状态有助于您监控服务环境的运行状况，排查配置问题，并确保可靠的作业处理。状态管理系统遵循计算环境中的既定模式，同时满足 SageMaker 培训作业集成的独特要求。

服务环境状态由 AWS Batch 基于配置验证、资源可用性和运行状况检查自动管理。与管理物理基础设施的计算环境不同，服务环境侧重于配置验证和与 SageMaker AI 服务的集成就绪。通过状态转换，您可以了解您的服务环境能否成功提交和管理 SageMaker 培训作业。

服务环境状态定义

服务环境可以处于四种可能的状态之一，这些状态表明其当前的运行状态和处理 SageMaker 培训作业的准备情况。每种状态都代表服务环境生命周期中的一个特定阶段，包括初始创建、运行就绪、最终删除等。下表描述了每种状态及其含义：

州	说明
CREATING	创建服务环境时的初始状态。在此状态下，AWS Batch 验证配置参数并建立与 SageMaker AI 服务的集成。服务环境无法处理作业，与之关联的任何作业队列都不会接受服务作业提交。对于正确配置的服务环境，创建过程通常会在几秒钟内完成。
VALID	表示服务环境已通过所有配置验证检查并准备好处理 SageMaker 训练作业的操作状态。此状态表示服务环境配置正确，所有必需的权限都已到位，并且 AWS Batch 可以代表您成功向 SageMaker AI 提交作业。服务环境的大部分运行生命周期都处于这一状态。
INVALID	一种状态，表示服务环境遇到了无法处理 SageMaker 训练作业的配置或权限问题。如果服务环境处于无效状态，则与该服务环境关联的

州	说明
	作业队列在相关问题得到解决之前将无法处理新的服务作业提交。
DELETING	请求删除服务环境时出现的状态。在此状态下， AWS Batch 确保没有活动的 T SageMaker raining 作业与环境相关联，并执行必要的清理操作。处于这种状态的服务环境无法处理新的作业提交，并且在所有关联的资源都已正确清理后，将立即完成删除过程。

服务环境状态变换

服务环境状态变换会根据配置更改、验证结果和运行状况监控自动进行。该 AWS Batch 服务持续监控服务环境的运行状况并相应地更新状态。了解这些变换有助于您预测配置更改何时生效，以及如何解决导致无效状态的问题。

成功完成创建和验证后，服务环境的状态将从 CREATING 变为 VALID。此过渡确认所有配置参数均正确，所需的 IAM 权限配置正确，服务环境可以成功与 SageMaker AI 服务集成。一旦进入 VALID 状态，关联的作业队列就可以开始处理服务作业提交。

当配置验证失败或依赖项不可用时，服务环境的状态会从 VALID 变为 INVALID。出现这种情况的可能原因包括 IAM 角色修改、违反配额的容量限制更改或影响服务环境运行能力的外部资源修改等。状态原因字段会提供有关导致无效状态的具体原因详情。

相关问题得到解决后，服务环境的状态可以从 INVALID 恢复到 VALID。这可能涉及更新 IAM 权限、更正容量配置或恢复对所需 AWS 资源的访问权限。通常，一旦 AWS Batch 检测到配置问题已得到解决，就会自动进行变换。

在 AWS Batch中创建服务环境

您需要先创建一个服务环境 AWS Batch，然后才能在中运行 SageMaker 训练作业。您可以创建包含与 SageMaker AI 服务集成所需的配置参数 AWS Batch 的服务环境，并代表您提交 SageMaker 训练作业。

先决条件

在创建服务环境之前，请确保您已满足下列前提条件：

- IAM 权限：创建和管理服务环境的权限。有关更多信息，请参阅 AWS Batch IAM 策略、角色和权限。

Create a service environment (AWS Console)

使用 AWS Batch 控制台通过 Web 界面创建服务环境。

创建服务环境

1. 打开 AWS Batch 控制台，网址为https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择环境。
3. 选择创建环境，然后选择服务环境。
4. 对于服务环境配置，请选择 SageMaker AI。
5. 对于名称，为您的服务环境输入一个唯一的名称。有效字符为 a-z、A-Z、0-9、连字符（-）和下划线（_）。
6. 对于最大实例数，请输入并发训练实例的最大数量
7. （可选）要添加标签，请选择添加标签并输入键值对。
8. 选择下一步。
9. 检查新服务环境的详细信息，然后选择创建服务环境。

Create a service environment (AWS CLI)

使用`create-service-environment`命令 AWS 通过 CLI 创建服务环境。

创建服务环境

1. 创建具有基本必需参数的服务环境：

```
aws batch create-service-environment \
    --service-environment-name my-sagemaker-service-env \
    --service-environment-type SAGEMAKER_TRAINING \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=10
```

2. （可选）创建带标签的服务环境：

```
aws batch create-service-environment \
    --service-environment-name my-sagemaker-service-env \
    --service-environment-type SAGEMAKER_TRAINING \
```

```
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=10 \
    --tags team=data-science,project=ml-training
```

3. 验证服务环境是否已成功创建：

```
aws batch describe-service-environments \
    --service-environment my-sagemaker-service-env
```

服务环境在“环境”列表中显示为 CREATING 状态。成功完成创建后，服务环境的状态将变为 VALID 并准备就绪，可以向其添加服务作业队列，从而可以开始处理作业。

更新中的服务环境 AWS Batch

您可以更新服务环境以修改其容量限制、更改其运行状态或更新资源标签。服务环境更新允许您在 SageMaker 培训工作负载要求变化时调整容量，或者修改操作设置，而无需重新创建环境。在更新服务环境之前，应了解可以修改哪些参数以及更改将对正在运行的作业产生的影响。

您可以更改服务环境的容量限制、状态或标签。

Update a service environment (AWS Console)

使用 AWS Batch 控制台通过 Web 界面更新服务环境。

更新服务环境

1. 打开 AWS Batch 控制台，网址为https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择环境。
3. 选择服务环境选项卡。
4. 选择要更新的服务环境。
5. 选择操作，然后选择以下任意一个选项：
   - 状态：选择启用或禁用以更改状态。
   - 容量限制：修改最大实例数
6. 选择保存更改以应用更改。

服务环境会立即更新。检查环境详细信息，以确认是否已成功应用更改。如果您禁用了服务环境，则在您将其重新启用之前，关联的作业队列将停止处理新的服务作业提交。

Update a service environment (AWS CLI)

使用update-service-environment命令 AWS 通过 CLI 修改服务环境。

更新服务环境容量限制

1. 更新服务环境的容量限制：

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --capacity-limits capacityUnit=NUM_INSTANCES,maxCapacity=20
```

2. 验证是否已成功应用更新：

```
aws batch describe-service-environments \
    --service-environments my-sagemaker-service-env
```

更新服务环境状态

1. 禁用服务环境以停止处理新作业：

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --state DISABLED
```

2. 重新启用服务环境以恢复处理：

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --state ENABLED
```

服务环境更新将立即生效。监控服务环境状态，以确保在提交新作业之前成功完成更新。

在 AWS Batch中删除服务环境

当您的 SageMaker 训练作业不再需要服务环境时，您可以将其删除。删除服务环境会移除配置并阻止进一步的作业提交。在删除服务环境之前，请确保没有活动的 SageMaker 训练作业依赖于该服务环境，并且没有作业队列与该服务环境相关联。

> **Important**
>
> 服务环境的删除是不可逆的。删除完成后，您将无法恢复服务环境或其配置。如果您将来需要类似的功能，则必须使用所需设置创建一个新的服务环境。如果您稍后可能需要重新激活该服务环境，可考虑将其禁用而不是删除。

> **Note**
>
> 删除账户中的所有服务环境不会自动移除为 AWS Batch 和 SageMaker AI 集成创建的服务相关角色。该服务相关角色仍然可用于在未来创建服务环境。如果要移除该服务相关角色，则必须在确保您的账户中不存在任何服务环境后，单独使用 IAM 将其删除。

删除的先决条件

要删除某个服务环境，您必须首先将所有服务作业队列取消关联，然后禁用该服务环境。

在删除服务环境之前：

- 检查活动作业-确保服务环境中当前没有正在运行的 SageMaker 训练作业。
- 检查作业队列：识别与该服务环境关联的作业队列，然后将作业队列关联到其他服务环境，或禁用并删除作业队列。

作业队列管理：与已删除的服务环境关联的作业队列可能会仍然存在，但无法处理服务作业。在删除原始服务环境之前，应首先删除未使用的作业队列或将其关联到其他服务环境。

Delete a service environment (AWS Console)

使用 AWS Batch 控制台通过 Web 界面删除服务环境。

删除服务环境

1. 打开 AWS Batch 控制台，网址为https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择环境。
3. 选择服务环境选项卡，然后选择一个服务环境。
4. 如果服务环境已启用，请选择操作，然后选择禁用。
5. 禁用服务环境后，选择操作，然后选择删除。

6. 在确认对话框中，选择确认。

删除期间，服务环境会显示 DELETING 状态。删除完成后，服务环境将从“环境”列表中消失。

Delete a service environment (AWS CLI)

使用`delete-service-environment`命令 AWS 通过 CLI 删除服务环境。

删除服务环境

1. 检查与该服务环境相关的作业队列：

```
aws batch describe-job-queues
```

   如果该服务环境关联了任何作业队列，则可以从该服务环境中将作业队列取消关联并将其关联到其他服务环境，也可以删除该作业队列。

2. 禁用服务环境：

```
aws batch update-service-environment \
    --service-environment my-sagemaker-service-env \
    --state DISABLED
```

3. 删除服务环境：

```
aws batch delete-service-environment \
    --service-environment my-sagemaker-service-env
```

4. 监控删除过程：

```
aws batch describe-service-environments \
    --service-environment my-sagemaker-service-env
```

在删除过程中，服务环境的状态将变为 DELETING。删除完成后，服务环境将不再在 describe 操作中列出。关联的作业队列会继续存在，但在关联到其他服务环境之前无法处理服务作业。

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。

主题

创建 Amazon EC2 作业队列

完成以下步骤，为 Amazon Elastic Compute Cloud（Amazon EC2）创建作业队列。

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。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 Fargate 作业队列

完成以下步骤，为 AWS Fargate 创建作业队列。

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。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 EKS 作业队列

完成以下步骤，为 Amazon Elastic Kubernetes Service（Amazon EKS）上创建作业队列。

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。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。

在 AWS Batch 中创建 SageMaker 训练作业队列

SageMaker 训练作业队列直接与 SageMaker AI 服务集成，无需管理底层计算基础设施，即可提供无服务器作业调度。

先决条件

在创建 SageMaker 训练作业队列之前，确保您已满足如下前提条件：

- 服务环境：一个定义了容量限制的服务环境。有关更多信息，请参阅 在 AWS Batch中创建服务环境。
- IAM 权限：创建和管理 AWS Batch 作业队列和服务环境的权限。有关更多信息，请参阅 AWS Batch IAM 策略、角色和权限。

Create a SageMaker Training job queue (AWS Batch console)

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择作业队列和创建。
3. 对于编排类型，选择 SageMaker 训练。
4. 对于作业队列配置：
   a. 对于名称，输入该作业队列的名称。
   b. 对于优先级，输入一个介于 0 到 1000 之间的值。服务环境会优先处理优先级较高的作业队列。
   c. （可选）对于计划策略 Amazon 资源名称（ARN），请选择现有的计划策略。
   d. 对于已连接的服务环境，从列表中选择一个要关联到该作业队列的服务环境。
5. （可选）对于 作业状态限制：
   a. 对于配置错误，选择 SERVICE_ENVIRONMENT_MAX_RESOURCE 并输入最大可运行时间（秒）。
   b. 对于容量，选择 INSUFFICIENT_INSTANCE_CAPACITY 并输入最大可运行时间（秒）。
6. 选择创建作业队列

Create a SageMaker Training job queue (AWS CLI)

使用 `create-job-queue` 命令创建 SageMaker 训练作业队列。

以下示例会创建一个使用服务环境的基本 SageMaker 训练作业队列：

```
aws batch create-job-queue \
  --job-queue-name my-sm-training-fifo-jq \
  --job-queue-type SAGEMAKER_TRAINING \
  --priority 1 \
  --service-environment-order order=1,serviceEnvironment=ExampleServiceEnvironment
```

将 *ExampleServiceEnvironment* 替换为服务环境的名称。

该命令返回的输出类似于下方内容：

```
{
```

```
  "jobQueueName": "my-sm-training-fifo-jq",
  "jobQueueArn": "arn:aws:batch:region:account:job-queue/my-sm-training-fifo-jq"
}
```

创建作业队列后，请验证该队列是否已成功创建且处于有效状态。

使用 describe-job-queues 命令查看有关作业队列的详细信息：

```
aws batch describe-job-queues --job-queues my-sm-training-fifo-jq
```

该命令返回的输出类似于下方内容：

```
{
  "jobQueues": [
    {
      "jobQueueName": "my-sm-training-fifo-jq",
      "jobQueueArn": "arn:aws:batch:region:account:job-queue/my-sm-training-fifo-
jq",
      "state": "ENABLED",
      "status": "VALID",
      "statusReason": "JobQueue Healthy",
      "priority": 1,
      "computeEnvironmentOrder": [],
      "serviceEnvironmentOrder": [
        {
          "order": 1,
          "serviceEnvironment": "arn:aws:batch:region:account:service-
environment/ExampleServiceEnvironment"
        }
      ],
      "jobQueueType": "SAGEMAKER_TRAINING",
      "tags": {},
      "jobStateTimeLimitActions": []
    }
  ]
}
```

请确保：

- state 为 ENABLED
- status 为 VALID

- statusReason 为 JobQueue Healthy
- jobQueueType 为 SAGEMAKER_TRAINING
- serviceEnvironmentOrder 会引用您的服务环境

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，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 参考中的CreateJobQueue。

> **Note**
>
> 您可以使用以下 AWS CLI 命令生成作业队列模板。
>
> ```
> $ aws batch create-job-queue --generate-cli-skeleton
> ```

```
{
   "computeEnvironmentOrder": [
      {
         "computeEnvironment": "",
         "order": 0
      }
   ],
   "jobQueueName": "",
   "jobStateTimeLimitActions": [
      {
         "state": "RUNNABLE",
         "action": "CANCEL",
   "maxTimeSeconds": 0,
         "reason": ""

      }
   ],
   "priority": 0,
   "schedulingPolicyArn": "",
   "state": "ENABLED",
   "tags": {
      "KeyName": ""
```

```
    }
}
```

在中查看作业队列 AWS Batch

创建作业队列并提交作业后，能够监控其进度非常重要。您可以使用作业详细信息页面来查看、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作业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。

作业队列详细信息

本部分提供了作业队列概述和维护选项。值得注意的是，您可以在本部分中找到 Amazon 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。

活跃股票

对于使用公平共享调度的任务队列， AWS Batch 提供对不同共享标识符如何消耗容量的可见性。此信息可帮助您了解资源分布并确定可能需要调整的份额。

> **Note**
>
> 只有当作业队列的调度算法为 Fair-Share 时，才会显示 “活动份额” 选项卡。

前 20 个活跃份额部分显示了其已计划、启动和正在运行的任务的共享标识符。此视图包括：

- 共享标识符名称-共享的唯一标识符。

  共享标识符是用于对作业进行分组以进行公平共享计划的标签。当您提交具有相同份额标识符的任务时，出于资源分配目的，会将其 AWS Batch 视为相同工作负载的一部分。共享标识符有助于确保计算能力在不同的团队、项目或工作负载类型之间公平分配。有关更多信息，请参阅 使用份额标识符标识工作负载。

- 容量利用率-配置作业要使用的资源量。这是以instancescpu、或vCPU根据环境来衡量的。
- 查看作业操作-用于查看该共享的所有作业的链接。

您可以以列表和图表格式查看此信息：

- 列表视图-表格显示精确的容量数字
- 图表视图-显示相对利用率的可视条形图

作业队列快照

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部分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。对于公平份额调度作业队列，作业的排序基于作业优先级和份额使用情况。所需容量字段显示配置作业要使用的资源量。

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以及其他相关信息。

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。

```
aws batch get-job-queue-snapshot --job-queue my-sm-training-fifo-jq
```

作业状态限制

使用此选项卡可以查看有关作业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

如果您的作业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。

删除 AWS Batch 中的作业队列

不再需要某个作业队列时，您可以将其禁用和删除。

Delete a job queue (AWS Batch console)

1. 打开AWS Batch控制台，地址：https://console.aws.amazon.com/batch/。
2. 在导航窗格中，选择作业队列，然后选择一个作业队列。
3. 选择操作，然后选择禁用。
4. 当作业队列的状态变为已禁用后，依次选择操作、删除。
5. 在模态窗口中，选择删除作业队列。

Delete a job queue (AWS CLI)

1. 禁用作业队列以防止提交新的作业：

```
aws batch update-job-queue \
  --job-queue my-sm-training-fifo-jq \
  --state DISABLED
```

2. 等待所有仍在运行的作业完成运行，然后删除该作业队列：

```
aws batch delete-job-queue \
  --job-queue my-sm-training-fifo-jq
```

公平份额调度策略

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 SubmitJob 中的 dependsOn 参数来指定每个作业的依赖关系。

如果您创建了调度策略并将其附加到作业队列，则公平份额调度将处于开启状态。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 使用调度策略分配份额标识符。

主题

使用份额标识符标识工作负载

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 *(T * weightFactor)* 公式来跟踪每个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。

您可以设置公平份额调度优先级，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。如果您未指定公平份额调度策略，则提交到该作业队列的所有作业都将按照 FIFO 顺序进行调度。您不能在提交作业时指定份额标识符或公平份额调度优先级。

> **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。

使用调度策略分配份额标识符

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。使用公平份额调度策略时，您可以为工作负载或用户分配不同的份额标识符。AWS Batch 将为每个公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，以备非活动份额标识符使用。有关更多信息，请参阅 SchedulingPolicyDetail。

使用公平份额调度来帮助调度作业

公平份额调度提供了一组控件来帮助调度作业。

> **Note**
>
> 有关调度策略参数的更多信息，请参阅 SchedulingPolicyDetail。

- 份额衰减秒数：AWS Batch 调度器用于计算每个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。预留比为 *(computeReservation/100)^ActiveFairShares*，其中 ActiveFairShares 是活动份额标识符的数量。

  > **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处理作业时，您可以通过作业队列快照查看前 100 个 RUNNABLE 作业的列表。有关更多信息，请参阅 在中查看作业队列 AWS Batch。

教程：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。创建公平份额调度策略时，您可以将一个或多个份额标识符或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）对于份额衰减秒数，为公平份额调度策略的份额衰减时间输入一个整数值。份额衰减时间越长，在调度作业时使用的时间将会越长，计算资源使用量也会显著增加。这样就可以在份额标识符最近没有使用计算资源时，让使用该份额标识符的作业使用的计算资源量暂时超过该份额标识符的权重所允许的计算资源量。
6. （可选）对于计算预留，为公平份额调度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份额标识符。

   预留比为 `(computeReservation/100)^ActiveFairShares`，其中 ActiveFairShares 是活动份额标识符的数量。

   例如将 `computeReservation` 的值设为 50 时，表示如果只有一个份额标识符，则 AWS Batch 应预留最大可用 VCPU 的 50%；如果有两个份额标识符，则应预留 25%；如果有三个份额标识符，则应预留 12.5%。将 `computeReservation` 的值设为 25 时，表示如果只有一个份额标识

符，则 AWS Batch 应预留最大可用 VCPU 的 25%；如果有两个份额标识符，则应预留 6.25%；如果有三个份额标识符，则应预留 1.56%。

7. 在份额属性部分中，您可以为要与公平份额调度策略关联的每个份额标识符指定份额标识符和权重。

   a. 选择添加份额标识符。

   b. 在份额标识符中，指定份额标识符。如果字符串以 '*' 结尾，则它将成为份额标识符前缀，用于匹配作业的份额标识符。调度策略中的所有份额标识符和份额标识符前缀都必须唯一，不能重叠。例如，您不能在同一个公平份额调度策略中使用份额标识符前缀“UserA*”和份额标识符“UserA1”。

   c. 在权重系数中，指定该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。如果使用份额标识符前缀，则份额标识符以该前缀开头的作业将共享该权重系数。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，请参阅标记 AWS Batch 资源。

9. 选择提交以完成并创建您的计划策略。

参考：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。

> **Note**
>
> 您可以使用以下 AWS CLI 命令生成作业队列模板。
>
> ```
> $ aws batch create-scheduling-policy --generate-cli-skeleton
> ```

```
{
    "name": "",
    "fairsharePolicy": {
        "shareDecaySeconds": 0,
        "computeReservation": 0,
```

```
        "shareDistribution": [
            {
                "shareIdentifier": "",
                "weightFactor": 0.0
            }
        ]
    },
    "tags": {
        "KeyName": ""
    }
}
```

资源感知调度

AWS Batch 根据作业队列（JQ）所关联计算环境（CE）中的 vCPU、GPU 和内存可用性来调度作业。但在某些时候，仅凭这些 CE 资源的可用性并不能保证作业能够成功运行，因为作业可能会依赖其他所需的资源，因此这些作业将被取消或终止。这会导致计算资源的使用效率低下。为解决此问题，资源感知调度可以在 CE 上调度作业运行之前检查所依赖的非 CE 资源的可用性。

AWS Batch 资源感知调度让您能够根据运行作业所需的消耗性资源（第三方许可证令牌、数据库访问带宽、限制对第三方 API 的调用的需求等）来调度作业。您可以指定作业运行所需的消耗性资源，Batch 在调度作业时会考虑这些资源依赖项。您可以避免进行手动干预，从而消除因消耗性资源短缺而导致的作业失败和长时间等待。您可以通过仅分配具有全部所需资源的作业来减少计算资源利用不足的情况。

资源感知调度支持 FIFO 和公平份额调度策略，并且可以与 Batch 支持的所有计算平台结合使用，包括 EKS、ECS 和 Fargate。此功能可用于数组作业、多节点并行（MNP）作业和常规 Batch 作业。

要配置资源感知调度，首先要指定运行作业所需的所有消耗性资源，以及每种资源的可用总数。然后，对于每项需要消耗性资源的作业，您可以指定所需每种资源的名称和所需数量。Batch 会跟踪作业队列中的作业有多少消耗性资源可用，并确保只有在作业成功运行所需的所有消耗性资源都可用时，才调度该作业运行。

主题

- 删除消耗性资源

创建消耗性资源

您必须首先创建消耗性资源，这些资源代表将在作业运行时消耗且数量有限的非 CE 资源。每种消耗性资源都有：

- 一个资源名称 (consumableResourceName)，该名称在账户级别必须唯一。
- （可选）资源类型 (resourceType)，指示作业完成后资源是否可供重复使用。这可以是以下类型之一：
  - REPLENISHABLE（默认值）
  - NON_REPLENISHABLE
- 总数量 (totalQuantity)，指定可用消耗性资源的总量。

每个账户的最大消耗性资源数量为 5 万个。

控制台：

1. 在 AWS Batch 控制台的左侧导航面板中，选择消耗性资源。
2. 选择创建消耗性资源。
3. 输入唯一的资源名称、资源总量，然后选择资源类型是可补充还是不可补充。
4. 选择创建消耗性资源。

API。

使用 CreateConsumableResourceAPI 来定义需要的资源。

指定运行作业所需的资源

注册作业时，您可以指定您创建的一个或多个资源的名称 (consumableResource) 以及该作业的每个实例所需的资源数量 (quantity)。

Batch 会跟踪每种资源在任何给定时刻的可用单位数。对于作业队列中的每个作业，Batch 调度器会确保您的作业仅在指定的资源依赖项可用时运行。

如果作业到达队列顶部时作业的消耗性资源不可用，则该作业将处于 RUNNABLE 状态等待，直到所有必需的资源都可用或达到作业状态时间限制为止（请参阅在中查看作业队列 AWS Batch）。一旦

Batch 验证所有资源都可用，作业状态就会变为 STARTING，然后变为 RUNNING。资源将在作业状态变为 STARTING 后立即锁定，然后在作业状态变为 SUCCEEDED 或 FAILED 时解锁。

您还可以在提交作业时更新特定作业所需的资源数量。

控制台：

要在定义作业时指定资源及其所需数量，请执行以下操作：

1. 使用 AWS Batch 控制台中的作业定义向导（作业定义 -> 创建）定义作业。
2. 在向导第 4 步：配置容器的消耗性资源下，从列表中选择所需资源的名称。在已请求的值字段中，输入此作业实例所需的此资源的数量，然后选择添加消耗性资源。
3. 为作业所需的所有消耗性资源重复上一步。您最多可以为定义的每个作业指定 5 个资源。
4. 在完成作业定义向导之后，但在选择创建作业定义之前，您将看到已创建的消耗性资源列表。

要在提交作业时更新所需资源数量，请执行以下操作：

1. 在 AWS Batch 控制台的导航窗格中，选择作业，然后选择提交新作业。
2. 在向导第 2 步：配置覆盖的消耗性资源覆盖下，针对要为作业覆盖所需数量的任何消耗性资源，输入一个新的已请求的值。
3. 完成要为该作业做出的所有覆盖后，选择下一步以继续执行审核并提交。

API。

将使用 RegisterJobDefinition API 注册作业时，请在请求的 consumableResourceProperties 部分中使用 consumableResourceList 来指定运行作业实例所需的消耗性资源以及每个消耗性资源的数量。

使用 SubmitJobAPI 提交作业时，可以使用请求的 consumableResourcePropertiesOverride 部分来覆盖消耗性资源列表和每个资源的数量。请注意，这只会覆盖每个作业实例所需的资源数量，而不是可用总量。

检查正在使用和可用的资源数量

Batch 允许您查询给定时刻的可用资源数量 (availableQuantity)、正在使用的资源数量 (inUseQuantity) 和总资源数量 (totalQuantity)。

作业进入 STARTING 状态后，将从该资源的可用数量中扣除已消耗的资源量。如果资源类型为 REPLENISHABLE，则在作业状态变为 SUCCEEDED 或 FAILED 后，已消耗的资源数量将立即加回到

可用数量中，但总数量保持不变。如果资源类型为 NON_REPLENISHABLE，则将从总量和可用数量中扣除已消耗的资源数量，无论作业状态是否变为 SUCCEEDED 或 FAILED 状态，都不会加回已消耗的资源数量。

> **Note**
>
> 此信息最长可能会延迟 30 秒。

控制台：

1. 在 AWS Batch 控制台的左侧导航面板中，选择消耗性资源。
2. 选择可补充或不可补充选项卡，查看您创建的该类型的资源。
3. 对于每种可补充的资源，控制台会显示名称、资源总量、当前正在使用的资源数量和仍然可用的资源数量，此外还会提高利用率的计算结果（正在使用的资源数量除以该资源的总量）。

   对于每种不可补充的资源，控制台都会显示名称、当前正在使用的数量以及仍然可用的数量。

您还可以从控制台的作业详细信息页面查看有关消耗性资源的当前信息。

1. 在 AWS Batch 控制台的左侧导航面板中，选择作业，然后选择作业名称以打开该作业的详细信息页面。
2. 如果作业需要，可以查看有关可补充资源和不可补充资源的信息。对于这两种类型，控制台都会显示资源的名称、该作业已请求的数量、仍然可用的数量、当前正在使用的数量、资源总量，以及当前利用率的计算结果（作业正在使用的资源数量除以该资源的总量）。

API。

使用 DescribeConsumableResource API，这将返回以下信息：

```
{
   "availableQuantity": number,
   "consumableResourceArn": "string",
   "consumableResourceName": "string",
   "createdAt": number,
   "inUseQuantity": number,
   "resourceType": "string",
   "tags": {
      "string" : "string"
```

```
    },
    "totalQuantity": number
}
```

在列出您在账户中创建的所有消耗性资源时，ListConsumableResources API 还会报告正在使用的资源数量 (inUseQuantity) 和当前可用的资源总数 (totalQuantity)。此 API 还允许您按消耗性资源名称筛选消耗性资源列表查询。

在作业使用资源时更新资源的数量

您可以将资源总量重置为某个新值，也可以从总量中加上或减去该值。

如果您指定的新总量大于之前的数量，Batch 会相应地调度更多作业。如果新总量少于之前的数量，并且该资源没有正在使用的单位，则 Batch 只会减少总量（或可用数量）。如果有正在使用的单位，Batch 会立即减少可用数量，并且在作业完成时，Batch 会减少总量（可用数量），使其最终等于该新数量。

控制台：

1. 在 AWS Batch 控制台的左侧导航面板中，选择消耗性资源。
2. 选择可补充或不可补充选项卡，查看您创建的该类型的资源。
3. 对于不可补充的资源：

    1. 选择要更新的资源，然后选择操作，再选择设置资源、添加资源或移除资源。
    2. 将显示一个弹出窗口，以便根据您在上一步中选择的操作，在其中设置总值、添加资源或移除资源。输入要设置为新总值的数量、要添加到总量中的数量或要从总量中减去的数量，然后选择确定。

    对于不可补充的资源：

    1. 选择要更新的资源，然后选择操作，再选择设置资源、添加资源或移除资源。
    2. 将显示一个弹出窗口，以便根据您在上一步中选择的操作，在其中设置可用值、添加资源或移除资源。输入要设置为新可用值、要添加到可用数量或要从可用数量中减去的数量，然后选择确定。

API。

使用 UpdateConsumableResource API 设置资源的新总量，或者增加或减少总量。

查找需要特定消耗性资源的作业

Batch 可让您检索需要特定消耗性资源的作业列表。

控制台：

1. 在 AWS Batch 控制台的左侧导航面板中，选择消耗性资源。
2. 从列表中，选择该消耗性资源的名称。这时将打开该此资源的详细信息页面。
3. 在搜索作业下，输入要应用到作业列表的任何筛选条件。您可以按作业名称（“等于”、“开头为”）、日期范围（作业创建时间）和其他条件（“作业队列”、“作业定义”、“共享作业标识符”）进行筛选。对于要应用的每个筛选条件类型，请从下拉列表中的可用选项中进行选择，然后输入所需的任何其他信息。

   选择搜索。
4. 将显示（筛选后的）需要消耗性资源的作业列表，包括作业的名称、状态、已请求的消耗性资源单位数、其他所需的消耗性资源等。使用此列表，您可以选择一个或多个要取消或终止的作业。也可以选择作业的名称来打开该作业的详细信息页面。
5. 在搜索作业下，您现在可以刷新结果或清除搜索并重新开始。

API。

您可以通过 `ListJobsByConsumableResource` API 获取使用特定消耗性资源的作业列表。此 API 还允许您按作业状态或作业名称筛选作业列表查询。

删除消耗性资源

您可以随时删除消耗性资源，即使需要该资源的作业仍在运行。删除消耗性资源后，从收到删除命令之时到作业调度器执行删除之时可能会有一段间隔，因此有可能消耗该资源的作业的调度时间恰好在 delete 调用之后。如果已删除消耗性资源的资源类型 (`resourceType`) 为 `REPLENISHABLE`，则这将在作业完成时忽略。如果您删除消耗性资源后使用相同的名称重新创建该资源，则该资源将被视为同一资源，并且可供 `RUNNABLE` 作业使用。

控制台：

1. 在 AWS Batch 控制台的左侧导航面板中，选择消耗性资源。
2. 选择可补充或不可补充选项卡，查看您创建的该类型的资源。
3. 选择要删除的每个资源，然后选择删除。这时将显示删除消耗性资源弹出窗口。要确认删除，请选择删除。

也可以从控制台的详细信息页面删除消耗性资源。

1. 在 AWS Batch 控制台的左侧导航面板中，选择消耗性资源。
2. 选择可补充或不可补充选项卡，查看您创建的该类型的资源。
3. 选择要删除的资源的名称。此时将显示该消耗性资源的详细信息页面。选择删除。这时将显示删除消耗性资源弹出窗口。要确认删除，请选择删除。

API。

使用 `DeleteConsumableResource` API 删除消耗性资源。

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一个作业定义，但可在运行时覆盖作业定义中指定的许多参数。

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。
- 容器要使用多少 v CPUs 和多少内存。
- 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。

内容

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（也称为分组调度）。

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

完成以下步骤，在 Amazon Elastic Compute Cloud（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。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，将命令的等效 JSON 字符串数组输入到该字段中。

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 Parameters。

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. 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（可选的）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 LogConfiguration: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 LinuxParameters: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. 在任务属性部分中：

a. 对于执行角色 – 条件，选择一个允许 Amazon ECS 代理代表您执行 AWS API 调用的角色。有关创建执行角色的更多信息，请参阅教程：创建 IAM 执行角色。

b. 选择启用 ECS execute 命令，以实现直接访问 Amazon ECS 容器 Shell 并绕过主机操作系统。必须选择一个任务角色。

> **Important**
>
> ECS execute 命令要求文件系统是可写的。

c. 对于任务角色，选择一个允许容器代表您执行 AWS API 调用的 Amazon ECS Identity and Access Management（IAM）角色。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 IAM 角色。

22. （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 LogConfiguration: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 LogConfiguration:secretOptions。

23. 选择下一页。

24. 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 Fargate 资源上创建单节点作业定义

完成以下步骤以在 AWS Fargate 资源上创建单节点作业定义。

要在 Fargate 资源上创建新作业定义，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，请参阅 Fargate 计算环境。
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。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，将命令的等效 JSON 字符串数组输入到该字段中。

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 Parameters。

c. （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

d. 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 安全代理会有一点儿内存开销。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，如何解决 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确定内存优先级。有关更多信息，请参阅 计算资源内存管理。

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. （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 LogConfiguration:secretOptions。

g. 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. 对于装载点配置，请选择添加装载点配置以添加数据卷的装载点。您必须指定源卷和容器路径。这些装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. 在任务属性部分中：

i. 对于执行角色 – 条件，选择一个允许 Amazon ECS 代理代表您执行 AWS API 调用的角色。有关创建执行角色的更多信息，请参阅教程：创建 IAM 执行角色。

ii. 选择启用 ECS execute 命令，以实现直接访问 Amazon ECS 容器 Shell 并绕过主机操作系统。必须选择一个任务角色。

> **Important**
>
> ECS execute 命令要求文件系统是可写的。

iii. 对于任务角色，选择一个允许容器代表您执行 AWS API 调用的 Amazon ECS Identity and Access Management（IAM）角色。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 IAM 角色。

h. 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 LogConfiguration: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 LogConfiguration: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

完成以下步骤，在 Amazon Elastic Kubernetes Service（Amazon EKS）上创建单节点作业定义。

要在 Amazon EKS 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）对于卷，选择添加卷，然后执行以下操作：

i. 为您的卷添加一个名称。

ii. （可选）添加主机该目录的主机路径。

iii. （可选）添加中等和大小限制来配置 Kubernetes emptyDir。

iv. （可选）提供容器组的密钥名称并选择该密钥是否为可选。

v. （可选）定义一个声明名称以将 Kubernetes 持久卷声明附加到该容器组，并选择该容器组是否为只读。

e. （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/`。

f. （可选）对于容器组注释，选择添加注释，然后输入名称-值对。

> **Important**
>
> 容器组标注的前缀不能包含 `kubernetes.io/`、`k8s.io/` 或 `batch.amazonaws.com/`。

g. 选择下一页。

h. 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。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）对于命令，输入要传递到容器的 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。

i. （可选）您可以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：

- 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。

j. 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。

k. （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。

l. （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。使用 SubPath 指定所引用卷内的子路径，而不是其根路径。

iii. 选择只读可删除该卷的写入权限。

iv. 选择添加卷装载。

m. （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。

n. （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。

o. （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。

p. （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。

q. （可选）开启以非根用户身份运行，从而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。

r. 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。

在 Amazon EC2 资源上创建使用多个容器的单节点作业定义

完成以下步骤，在 Amazon Elastic Compute Cloud（Amazon EC2）资源上创建具有多个容器的单节点作业定义。

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. 选择创建。
5. 对于编排类型，选择 Amazon Elastic Compute Cloud (Amazon EC2)。
6. 对于作业定义结构，请禁使用传统的 containerProperties 结构处理。
7. 对于 EC2 平台配置，请关闭启用多节点并行处理。
8. 选择下一步。
9. 在常规配置部分中，输入以下内容：

   a. 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（_）。

   b. 对于执行超时 – 可选，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。

   c. 开启调度优先级 – 可选。输入介于 0 到 100 之间的计划优先级值。值越高，优先级越高。

d. 展开标签 – 可选，然后选择添加标签以向该资源添加标签。输入键和可选的值，然后选择添加标签。

e. 开启传播标签，以将标签从作业和作业定义传播到 Amazon ECS 任务。

10. 在重试策略 – 可选部分中，输入以下内容：

a. 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。

b. 对于重试策略条件，选择添加退出时评估。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. 在任务属性部分中，输入以下内容：

a. 对于执行角色 – 条件，选择一个允许 Amazon ECS 代理代表您执行 AWS API 调用的角色。有关创建执行角色的更多信息，请参阅教程：创建 IAM 执行角色。

b. 选择启用 ECS execute 命令，以实现直接访问 Amazon ECS 容器 Shell 并绕过主机操作系统。必须选择一个任务角色。

> **Important**
>
> ECS execute 命令要求文件系统是可写的。

c. 对于任务角色，选择一个允许容器代表您执行 AWS API 调用的 Amazon ECS Identity and Access Management（IAM）角色。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 IAM 角色。

d. 对于 IPC 模式，选择 `host`、`task` 或 `none`。如果指定了 `host`，则在同一容器实例上指定了主机 IPC 模式的任务中的所有容器将与主机 Amazon EC2 实例共享相同的 IPC 资源。如果指定了任务，则指定任务中的所有容器将共享相同的 IPC 资源。如果指定了任何选项，则任务

的容器中的 IPC 资源是私有的，不与任务中或容器实例上的其他容器共享。如果未指定任何值，则 IPC 资源命名空间共享取决于容器实例上的 Docker 守护程序设置。

e. 对于 PID 模式，选择 `host` 或 `task`。例如，监控 sidecar 可能需要 `pidMode` 访问有关在同一任务中运行的其他容器的信息。如果指定了 `host`，则在同一容器实例上指定了主机 PID 模式的任务中的所有容器将与主机 Amazon EC2 实例共享相同的进程命名空间。如果指定了 `task`，则指定任务中的所有容器将共享相同的过程命名空间。如果未指定任何值，则默认值为每个容器的私有命名空间。

12. 在消耗性资源部分中，输入以下内容：

    a. 输入唯一的名称和已请求的值。

    b. 您可以通过选择添加消耗性资源来添加更多消耗性资源。

13. 在存储部分中，输入以下设置：

    a. 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。

    b. 您可以通过选择添加卷来添加更多卷。

14. 对于参数，选择添加参数以添加参数替换占位符，作为键（可选的）值对。

15. 选择下一页。

16. 在容器配置部分：

    a. 对于 Name（名称），输入容器的名称。

    b. 对于基本容器，如果该容器为必需，则启用该容器。

    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。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。

d. 对于资源要求，配置以下各项参数：

   i. 对于 vCPU 数，选择容器的 CPU 数量。

   ii. 对于内存，选择容器的内存量。

   iii. 对于 GPU – 可选，选择该容器 GPU 数量。

e. 对于 User，输入要在容器内使用的用户名。

f. 开启启用只读文件系统以移除对卷的写入权限。

g. 开启特权，以在主机实例上授予作业容器升级权限，类似于根用户。

h. 对于命令，将命令的等效 JSON 字符串数组输入到该字段中。

   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 Parameters。

i. 对于存储库凭证 – 可选，输入包含您的凭证的密钥的 ARN。

j. 对于环境变量 可选，选择添加环境变量以添加要传递到容器的环境变量。

k. 在 Linux 参数 – 可选部分中：

   i. 开启启用初始化进程以在容器内运行初始化进程。

   ii. 对于共享内存大小，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。

v. 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。

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

l.

> **Note**
>
> Firelens 日志记录必须在专用容器中进行。配置 Firelens 日志记录：
>
> - 在除专用 firelens 容器之外的每个容器中，将日志记录驱动程序设置为 `awsfirelens`
> - 在您的 Firelens 容器中，将 Firelens 配置 – 可选和日志记录配置 – 可选设置为日志记录的目标

在 Linux 配置 – 可选部分中：

> **Important**
>
> AWS Batch 会在非 MNP、非 Fargate 的 Amazon ECS 作业上强制执行 `host` 网络模式。Amazon ECS Firelen 必须使用根用户。运行使用 `host` 网络模式的作业时，Amazon ECS 建议不要使用根用户（UID 0）运行容器，以提供安全性。因此，所有使用 Firelens 日志记录的非 MNP、非 Fargate ECS 作业都不符合安全最佳实践。

i. 对于类型，选择 `fluentd`或 `fluentbit`。

ii. 对于选项，输入该选项的名称/值对。您可以使用已添加的选项来添加其他选项。

m. 在日志记录配置 – 可选部分中：

i. 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 LogConfiguration:logDriver。

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

ii. 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. 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 LogConfiguration:secretOptions。

n. 对于挂载点 – 可选，选择添加挂载点以添加数据卷的挂载点。您必须指定源卷和容器路径。

o. 对于密钥 – 可选，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 LogConfiguration:secretOptions。

p. 对于 Ulimits – 可选，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。

q. 对于依赖项 – 可选，选择添加容器依赖项。选择容器名称，其状态决定了此容器何时会启动。

17. 如果只配置了一个容器，则必须选择添加容器并完成新容器的配置。否则，请选择下一步进行检查。

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。本主题专门介绍如何为 AWS Batch 多节点并行作业创建作业定义（也称为分组调度）。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。

内容

教程：在 Amazon EC2 资源上创建多节点并行作业定义

在 Amazon Elastic Compute Cloud（Amazon EC2）资源上创建多节点并行作业定义。

> **Note**
>
> 要创建单节点作业定义，请参阅 在 Amazon EC2 资源上创建单节点作业定义。

要在 Amazon EC2 资源上创建多节点并行作业定义，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. 在导航窗格中，选择作业定义。
4. 选择创建。
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（可选的）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。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，将命令的等效 JSON 字符串数组输入到该字段中。

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 Parameters。

e. 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。

f. 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。

g. （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。

h. （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。

i. （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 LogConfiguration: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 LinuxParameters: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. 在任务属性部分中：

a. 对于执行角色 – 条件，选择一个允许 Amazon ECS 代理代表您执行 AWS API 调用的角色。有关创建执行角色的更多信息，请参阅教程：创建 IAM 执行角色。

b.

> **Important**
>
> 要使用 ECS execute 命令，您的计算环境必须满足多节点并行作业的计算环境注意事项部分的说明。

选择启用 ECS execute 命令，以实现直接访问 Amazon ECS 容器 Shell 并绕过主机操作系统。必须选择一个任务角色。

> **Important**
>
> ECS execute 命令要求文件系统是可写的。

c. 对于任务角色，选择一个允许容器代表您执行 AWS API 调用的 Amazon ECS Identity and Access Management（IAM）角色。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 IAM 角色。

27. （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 LogConfiguration: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 LogConfiguration:secretOptions。

28. 选择下一页。

29. 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 ContainerProperties 的作业定义模板

以下是一个空的作业定义模板，其中包含一个容器。您可以使用此模板创建任务定义，随后可将任务定义保存到文件并与 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 JobDefinition。

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

```
{
    "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
                    }
                }
            ]
        }
    }
}
```

ContainerProperties 的作业定义参数

使用 `ContainerProperties` 的作业定义分为几个部分：

- 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，请参阅创建多节点并行作业定义。

类型：字符串

有效值：`container` | `multinode`

必需：是

Parameters

`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。这种命名惯例是为AWS Batch服务所设置的变量保留的。

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 IAM 执行角色。

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：FargatePlatformConfiguration 对象

必需：否

`platformVersion`

作业所使用的AWS Fargate 平台版本，或LATEST使用最近批准的AWS Fargate 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》的中的实例存储交换卷或如何使用交换文件分配内存以便在 Amazon EC2 实例中用作交换空间？。

- 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目前支持提供给 Docker 进程守护程序的一小部分日志记录驱动程序（在 LogConfiguration 数据类型中显示）。

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会启用`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目前支持提供给 Docker 进程守护程序的一小部分日志记录驱动程序。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 Logs 日志记录驱动程序。有关更多信息，请参阅使用 awslogs 日志驱动程序和 Docker 文档中的 Amazon CloudWatch Logs 日志记录驱动程序。

`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**
>
> 如果有之前未列出但您希望与 Amazon ECS 容器代理一起使用的自定义驱动程序，则可使 GitHub 上提供的 Amazon ECS 容器代理项目分支，并对其进行自定义以便与该驱动程序结合使用。我们鼓励您针对要包含的更改提交拉取请求。但是，亚马逊云科技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 参数存在于要启动的任务所在的同一AWS 区域中，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。

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

默认值：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
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 参数存在于要启动的作业所在的同一AWS 区域中，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。

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：EksPodProperties 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。这种命名惯例是为AWS Batch所设置的变量保留的。

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。长度上限为 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。长度上限为 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 RegisterJobDefinition API 操作中没有为 dnsPolicy 指定任何值，那么 DescribeJobDefinitions 或 DescribeJobs API 操作都不会为 dnsPolicy 返回任何值。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工作负载为仅出口，不需要为传入连接的每个容器组分配 IP 的开销。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：EksEmptyDir 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：EksHostPath 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

名称

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

optional

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：NodeRangeProperty 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将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：EvaluateOnExit 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 标记 AWS Batch 资源。

类型：字符串到字符串映射

必需：否

Timeout

`timeout`

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。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将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 EcsProperties 创建作业定义

通过使用 `EcsProperties` 的 AWS Batch 作业定义，您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。您可以使用此功能从逻辑上整理工作负载组件，并将它们与主应用程序分

开。此功能可以与 Amazon Elastic Container Service（Amazon ECS）、Amazon Elastic Kubernetes Service（Amazon EKS）和 AWS Fargate 上的 AWS Batch 结合使用。

ContainerProperties 与 EcsProperties 作业定义对比

您可以根据应用场景的规定选择使用 ContainerProperties 或 EcsProperties 作业定义。从较高层次来看，使用 EcsProperties 运行 AWS Batch 作业与使用 ContainerProperties 运行作业类似。

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

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

例如，在 Amazon ECS 和 Fargate 上使用 ContainerProperties 的作业定义具有以下结构：

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

在 Amazon ECS 和 Fargate 上使用 EcsProperties 的作业定义具有以下结构：

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

对 AWS Batch API 的一般更改

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

- ContainerProperties 内所使用的许多参数出现在 TaskContainerProperties 中。一些示例包括 command、image、privileged、secrets 和 users。可以在 TaskContainerProperties 中找到它们。
- 有些 TaskContainerProperties 参数在旧版结构中没有功能等效项。一些示例包括 dependsOn、essential、name、ipcMode 和 pidMode。有关更多信息，请参阅 EcsTaskDetails 和 TaskContainerProperties。

  此外，有些 ContainerProperties 参数在 EcsProperties 结构中没有等效项或应用。在 taskProperties 中，container 已替换为 containers，因此新对象最多可以接受十个元素。欲了解更多信息，请参阅 RegisterJobDefinition:containerProperties 和 EcsTaskProperties:containers。
- taskRoleArn 在功能上与 jobRoleArn 等效。欲了解更多信息，请参阅 EcsTaskProperties:taskRoleArn 和 ContainerProperties:jobRoleArn。
- 您可以在 EcsProperties 结构中包含一（1）到十（10）个容器。有关更多信息，请参阅 EcsTaskProperties:containers。
- taskProperties 和 instanceTypes 对象是数组，但目前只接受一个元素。例如 EcsProperties:taskProperties 和 NodeRangeProperty:instanceTypes。

Amazon ECS 的多容器作业定义

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

- ecsProperties 与单容器定义中的 containerProperties 级别相同。有关更多信息，请参阅《AWS Batch API 参考指南》中的 EcsProperties。
- 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》中的 EcsProperties。
- containers 包含与单容器定义中的 containerProperties 类似的信息。主要区别在于，containers 使您可以最多定义十个容器。有关更多信息，请参阅《AWS Batch API 参考指南》中的 ECSTaskProperties:containers。
- essential 参数指示容器如何影响作业。所有关键容器都必须成功完成（以 0 退出），作业才能继续进行。如果标记为关键的容器失败（以非 0 退出），则作业将会失败。

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项列表。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作业的开始时间。如果依赖项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 ecsProperties 的 RegisterJobDefinition 请求语法。

Amazon EKS 的多容器作业定义

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

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

  这些容器已在 Amazon EKS Connector 代理中注册，并将注册信息保存在 Amazon Elastic Kubernetes Service 后端数据存储中。`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 Init 容器。

> **Note**
>
> `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，则作业将保持 `STARTING` 状态，直到它们完成。

- `shareProcessNamespace` 指示容器组（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 `true` 可以让容器看到位于同一容器组（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），作业才能成功。如果一个容器失败（以非 0 退出），则作业将失败。

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

参考：使用 EcsProperties 的 AWS Batch 作业场景

为了说明如何根据您的需求来构造使用 EcsProperties 的 AWS Batch 作业定义，本主题介绍了以下 RegisterJobDefinition 有效载荷。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）调用 RegisterJobDefinition。

Amazon EC2 上的 Amazon ECS 的 AWS Batch 作业

以下是 Amazon Elastic Compute Cloud 上的 Amazon Elastic Container Service 的 AWS Batch 作业示例：

```
{
    "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": false,
```

```
            "command": [
              "echo",
              "hello world"
            ],
            "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
            "resourceRequirements": [
              {
                "type": "VCPU",
                "value": "2"
              },
              {
                "type": "MEMORY",
                "value": "4096"
              }
            ]
          },
          {
            "name": "c3",
            "essential": true,
            "command": [
              "echo",
              "hello world"
            ],
            "image": "public.ecr.aws/amazonlinux/amazonlinux:latest",
            "firelensConfiguration": {
              "type": "fluentbit",
              "options": {
                "enable-ecs-log-metadata": "true"
              }
             },
            "resourceRequirements": [
              {
                "type": "VCPU",
                "value": "6"
              },
              {
                "type": "MEMORY",
                "value": "12288"
              }
            ]
          }
        ]
      }
    ]
```

```
  }
}
```

Fargate 上的 Amazon ECS 的 AWS Batch 作业

以下是 AWS Fargate 上的 Amazon Elastic Container Service 的 AWS Batch 作业示例：

```
{
    "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"
          }
        ]
  }
}
```

适用于 Amazon EKS 的 AWS Batch 作业

以下是 Amazon Elastic Kubernetes Service 的 AWS Batch 作业示例：

```
{
  "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"
            }
          }
```

```
        }
      ]
    }
  }
}
```

每个节点具有多个容器的 MNP AWS Batch 作业

以下是每个节点具有多个容器的多节点并行（MNP）AWS Batch 作业的示例：

```
{
  "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。

AWS Batch JobDefiniton 数据类型中的 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

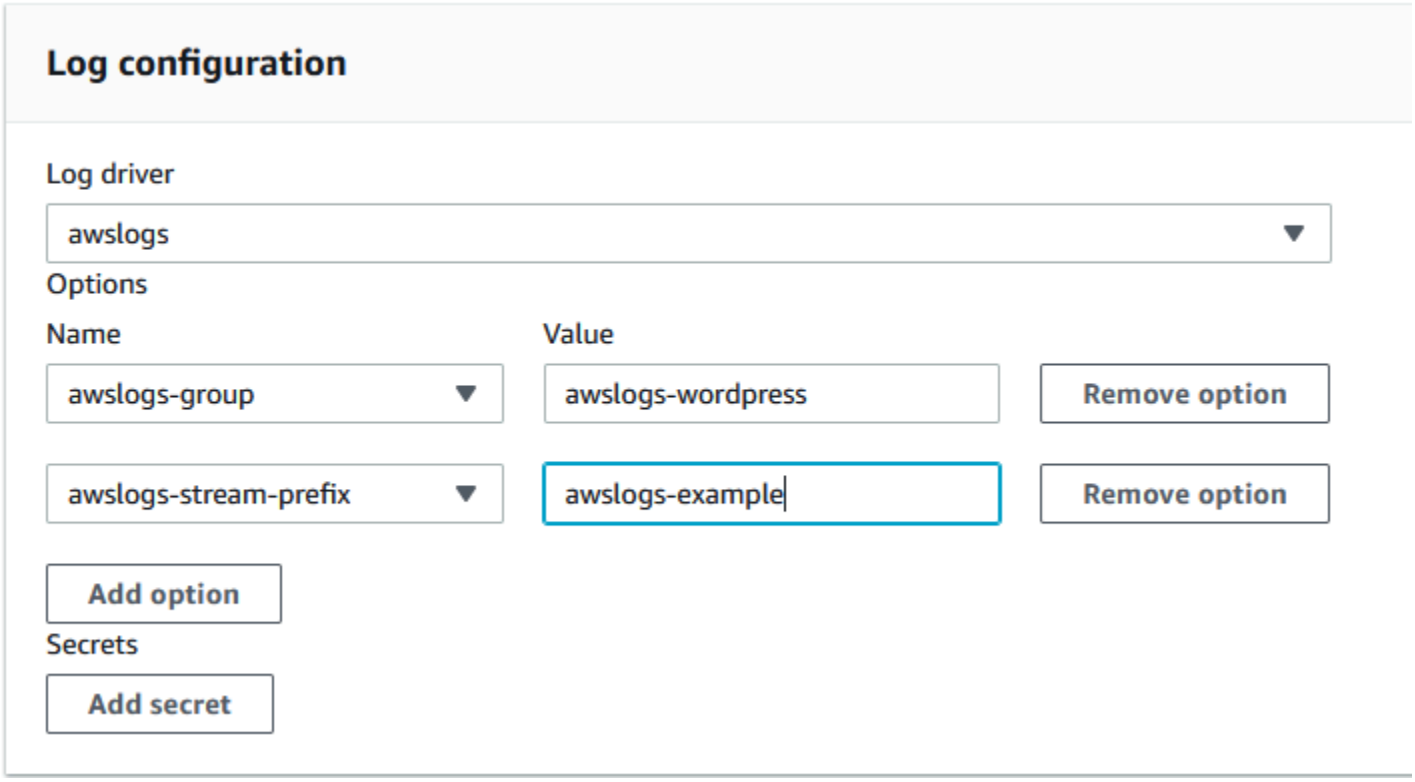
默认情况下，AWS Batch会启用`awslogs`日志驱动程序。本节介绍如何为作业自定义`awslogs`日志配置。有关更多信息，请参阅 创建单节点作业定义。

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 教程：提交作业。

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 密钥或 Paramet AWS Systems Manager er Store 参数中，然后在作业定义中引用它们。

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 指定敏感数据
- 使用 Systems Manager Parameter Store 指定敏感数据

使用 Secrets Manager 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。

使用 Secrets Manager 指定敏感数据时的注意事项

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。

  要将密钥的完整内容注入为环境变量或将密钥注入日志配置中，您的容器实例必须具有版本 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 IAM 执行角色。

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-2:777777777777:secret:<secret_name>",
                "arn:aws:kms:us-east-2:777777777777:key/<key_id>"
            ]
        }
    ]
}
```

作为环境变量注入敏感数据

您可以在作业定义中指定以下项目：

- `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 IDs 用于在密钥的不同版本更新或轮换时对其进行跟踪。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

以下显示了 > get-secret-value 命令的输出示例，该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"
    }]
  }]
}
```

在日志配置中注入敏感数据

在作业定义中指定 `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. 打开 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. 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 版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 IAM 执行角色。

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：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters",
                "secretsmanager:GetSecretValue",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:ssm:us-east-2:999999999999:parameter/<parameter_name>",
                "arn:aws:secretsmanager:us-
east-2:999999999999:secret:<secret_name>",
                "arn:aws:kms:us-east-2:999999999999: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"
    }]
  }]
}
```

在日志配置中注入敏感数据

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**
>
> 如果您的作业定义引用了存储在 Amazon ECR 中的映像，则此主题不适用。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。

对于 Fargate 上托管的作业，此功能需要平台版本 `1.2.0` 或更高版本。有关信息，请参阅《Amazon Elastic Container Service 开发人员指南》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 管理控制台，必须始终指定密钥的完整 ARN。

下面是显示必需参数的作业定义代码段：

```
"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

使用此功能需要执行角色。这允许容器代理拉取容器映像。有关更多信息，请参阅 AWS Batch IAM 执行角色。

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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:secret_name",
                "arn:aws:kms:us-east-1:123456789012:key/key_id"
            ]
        }
    ]
}
```

教程：创建私有注册表身份验证的密钥

完成以下步骤，使用为您的私有注册表凭证创建密钥 AWS Secrets Manager。

创建基本密钥

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 IAM 执行角色的更多信息，请参阅 AWS Batch IAM 执行角色。

作业定义示例

以下主题中的作业定义示例展示了如何使用环境变量、参数替换和卷挂载等常用模式。

内容

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://amzn-s3-demo-source-bucket/myjob.sh"
            },
            {
                "name": "BATCH_FILE_TYPE",
                "value": "script"
            }
        ],
        "user": "nobody"
    }
}
```

参数替换

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 Parameters。

```
{
    "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

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 JobDefinition。

您可以提交大量独立的简单作业。

主题

教程：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. 对于命令，将命令的等效 JSON 字符串数组输入到该字段中。

    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 refer https://docs.docker.com/engine/ence/builder/ #cmd。

    > **Note**
    > 此参数不能包含空字符串。

    b. 对于 v CPUs，输入CPUs 要为容器保留的 v 数。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“数量”中 GPUs，选择 GPUs 要为集装箱保留的数量。

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。

中的服务职位 AWS Batch

AWS Batch 服务作业使您能够通过 AWS Batch 任务队列向 AWS 服务提交请求。目前， AWS Batch 支持将 SageMaker 训练作业作为服务作业。与 AWS Batch 管理底层容器执行的容器化作业不同，服务作业 AWS Batch 允许提供作业调度和排队功能，而目标 AWS 服务（例如 SageMaker AI）则处理实际的任务执行。

AWS Batch fo SageMaker r Training 作业允许数据科学家将具有优先级的训练作业提交到可配置队列，从而确保工作负载在资源可用时立即在没有干预的情况下运行。此功能可解决资源协调、防止意外超支、满足预算约束、使用预留实例优化成本等常见挑战，以及无需在团队成员之间进行手动协调。

服务作业与容器化作业有多个关键的区别：

- 任务提交：必须使用 SubmitServiceJobAPI 提交服务作业。无法通过 AWS Batch 控制台提交服务作业。
- 任务执行： AWS Batch 计划和排队服务作业，但目标 AWS 服务运行实际的作业工作负载。
- 资源标识符：服务作业使用 ARNs 包含 “服务作业” 而不是 “作业” 的任务来区分它们与容器化作业。

要开始使用 SageMaker 培训 AWS Batch 服务作业，请参阅the section called “ SageMaker 人工智能 AWS Batch 入门”。

主题

中的服务任务有效负载 AWS Batch

使用提交服务作业时 SubmitServiceJob，您需要提供两个用于定义任务的关键参数：serviceJobType、和serviceRequestPayload。

- serviceJobType指定哪个 AWS 服务将执行作业。对于 SageMaker 培训作业，此值为SAGEMAKER_TRAINING。
- serviceRequestPayload 是一个 JSON 编码的字符串，其中包含通常会直接发送到目标服务的完整请求。对于 SageMaker 训练作业，此负载包含的参数与您在 SageMaker A CreateTrainingJobI API 中使用的参数相同。

有关所有可用参数及其描述的完整列表，请参阅 SageMaker A CreateTrainingJobI API 参考。CreateTrainingJob 支持的所有参数都可以包含在服务作业有效载荷中。

有关更多训练作业配置的示例 APIs，请参阅 CLI 和 SDKs SageMaker AI 开发者指南。

我们建议使用 PySDK 创建服务作业，因为 PySDK 提供了帮助程序类和实用程序。有关使用 pySDK 的示例，请参阅上的 SageMaker AI 示例。 GitHub

服务作业有效载荷示例

以下示例显示了运行 “hello world” SageMaker 训练脚本的训练作业的简单服务作业负载：

调用 SubmitServiceJob 时，此有效载荷将作为 JSON 字符串传递给 serviceRequestPayload 参数。

```
{
  "TrainingJobName": "my-simple-training-job",
  "RoleArn": "arn:aws:iam::123456789012:role/SageMakerExecutionRole",
  "AlgorithmSpecification": {
    "TrainingInputMode": "File",
    "TrainingImage": "763104351884.dkr.ecr.us-west-2.amazonaws.com/pytorch-
training:2.0.0-cpu-py310",
    "ContainerEntrypoint": [
      "echo",
      "hello world"
    ]
  },
  "ResourceConfig": {
    "InstanceType": "ml.c5.xlarge",
    "InstanceCount": 1,
```

```
    "VolumeSizeInGB": 1
  },
  "OutputDataConfig": {
    "S3OutputPath": "s3://your-output-bucket/output"
  },
  "StoppingCondition": {
    "MaxRuntimeInSeconds": 30
  }
}
```

在中提交服务作业 AWS Batch

要向提交服务任务 AWS Batch，请使用 SubmitServiceJobAPI。您可以使用 AWS CLI 或 SDK 提交作业。

如果您还没有执行角色，则必须首先创建一个，然后才能提交服务作业。要创建 SageMaker AI 执行角色，请参阅 SageMaker AI 开发者指南中的如何使用 SageMaker AI 执行角色。

服务作业提交工作流

提交服务任务时，请 AWS Batch 遵循以下工作流程：

1. AWS Batch 接收您的SubmitServiceJob请求并验证 AWS Batch特定参数。传递 serviceRequestPayload 但不进行验证。
2. 作业进入 SUBMITTED 状态并放入指定的作业队列中
3. AWS Batch 评估服务环境中是否有可用容量用于排在RUNNABLE队列前面的作业
4. 如果容量可用，则任务将移至SCHEDULED，任务已传递给 SageMaker AI
5. 获得容量并且 SageMaker AI 下载了服务作业数据后，服务作业将开始初始化并将任务更改为STARTING。
6. 当 SageMaker AI 开始执行任务时，其状态将更改为RUNNING。
7. 当 SageMaker AI 执行任务时， AWS Batch 监控其进度并将服务状态映射到 AWS Batch 作业状态。有关如何映射服务作业状态的详细信息，请参阅???
8. 当服务作业完成后，其状态将变为 SUCCEEDED 并且所有输出都可随时下载。

先决条件

提交服务作业之前，请确保您满足以下条件：

- 服务环境：一个定义了容量限制的服务环境。有关更多信息，请参阅 在 AWS Batch中创建服务环境。
- SageMaker 作业队列-用于提供 SageMaker 作业调度的作业队列。有关更多信息，请参阅 在 AWS Batch 中创建 SageMaker 训练作业队列。
- IAM 权限：创建和管理 AWS Batch 作业队列和服务环境的权限。有关更多信息，请参阅 AWS Batch IAM 策略、角色和权限。

使用 AWS CLI 提交服务作业

以下说明如何使用 AWS CLI 提交服务作业：

```
aws batch submit-service-job \
    --job-name "my-sagemaker-training-job" \
    --job-queue "my-sagemaker-job-queue" \
    --service-job-type "SAGEMAKER_TRAINING" \
    --service-request-payload '{\"TrainingJobName\": \"sagemaker-training-job-
example\", \"AlgorithmSpecification\": {\"TrainingImage\": \"123456789012.dkr.ecr.us-
east-1.amazonaws.com/pytorch-inference:1.8.0-cpu-py3\", \"TrainingInputMode
\": \"File\", \"ContainerEntrypoint\":  [\"sleep\", \"1\"]}, \"RoleArn\":
\"arn:aws:iam::123456789012:role/SageMakerExecutionRole\", \"OutputDataConfig\":
 {\"S3OutputPath\": \"s3://example-bucket/model-output/\"}, \"ResourceConfig\":
 {\"InstanceType\": \"ml.m5.large\", \"InstanceCount\": 1, \"VolumeSizeInGB\": 1}}'
    --client-token "unique-token-12345"
```

有关 `serviceRequestPayload` 参数的更多信息，请参阅 the section called “服务作业有效载荷”。

将 AWS Batch 服务作业状态映射到 SageMaker AI 状态

使用向 SageMaker 作业队列提交作业时 SubmitServiceJob，会 AWS Batch 管理作业生命周期并将 AWS Batch 作业状态映射到等效的 SageMaker 训练作业状态。服务作业（例如 SageMaker 训练作业）遵循的状态生命周期与传统的容器作业不同。服务作业的大多数状态与容器作业相同，不过服务作业引入了 SCHEDULED 状态并具有不同的重试行为，尤其是在处理目标服务容量不足错误时。

下表显示了 AWS Batch 作业状态和相应的 SageMaker状态/SecondaryStatus：

Batch 状态	SageMaker AI 主要状态	SageMaker AI 二级状态	说明
SUBMITTED	不适用	不适用	作业已提交到队列，等待调度器评估。
RUNNABLE	不适用	不适用	作业已排队，可以进行调度。一旦服务环境中有足够的资源可用，处于此状态的作业就会立即启动。如果没有足够的资源可用，作业会无限期地保持此状态。
SCHEDULED	InProgress	Pending	服务作业已成功提交给 SageMaker AI
STARTING	InProgress	Downloading	SageMaker 下载数据和图像的训练作业。已取得训练作业容量，并且已开始作业初始化。
RUNNING	InProgress	Training	SageMaker 训练作业执行算法
RUNNING	InProgress	Uploading	SageMaker 训练作业在训练完成后上传输出工件
SUCCEEDED	Completed	Completed	SageMaker 训练作业成功完成。已完成输出构件上传。
FAILED	Failed	Failed	SageMaker 训练作业遇到了一个不可恢复的错误。
FAILED	Stopped	Stopped	SageMaker 已使用手动停止训练作业`StopTrainingJob` 。

中的服务作业重试策略 AWS Batch

服务作业重试策略 AWS Batch 允许在特定条件下自动重试失败的服务作业。

由于多方面的原因，服务作业可能需要多次尝试：

- 临时服务问题：内部服务错误、节流或临时中断可能会导致作业在提交或执行期间失败。

- 训练初始化失败：作业启动期间出现的问题（例如映像提取问题或初始化错误）可能会在重试后得到解决。

通过配置适当的重试策略，可以提高作业成功率并减少手动干预需要，尤其是对于长时间运行的训练工作负载。

> **Note**
>
> 服务作业会自动重试某些类型的失败（例如容量不足错误），而不会消耗您配置的重试次数。您的重试策略主要处理其他类型的失败，例如算法错误或服务问题。

配置重试策略

使用配置服务作业重试策略 ServiceJobRetryStrategy，它支持简单的重试计数和有条件的重试逻辑。

重试配置

最简单的重试策略会指定服务作业失败后应进行的重试次数：

```
{
  "retryStrategy": {
    "attempts": 3
  }
}
```

此配置允许在服务作业失败后最多重试 3 次。

> **Important**
>
> `attempts` 值表示可以将作业置于 `RUNNABLE` 状态的总次数，包括首次尝试。值为 3 表示该作业将首次尝试一次，如果失败，则最多可再重试 2 次。

使用重试配置 evaluateOnExit

您可以使用 `evaluateOnExit` 参数来指定应重试或放任作业失败的条件。当不同类型的失败需要不同的处理方式时，这将非常实用。

evaluateOnExit 数组最多可以包含 5 个重试策略，每种策略都根据状态原因指定一个操作（RETRY 或 EXIT）以及条件：

```
{
  "retryStrategy": {
    "attempts": 5,
    "evaluateOnExit": [
      {
        "action": "RETRY",
        "onStatusReason": "Received status from SageMaker: InternalServerError*"
      },
      {
        "action": "EXIT",
        "onStatusReason": "Received status from SageMaker: ValidationException*"
      },
      {
        "action": "EXIT",
        "onStatusReason": "*"
      }
    ]
  }
}
```

此配置：

- 重试由于 SageMaker AI 内部服务器错误而失败的作业
- 立即让遇到验证异常（无法通过重试解决的客户端错误）的作业失败
- 包括适用于任何其他故障失败类型的 catch-all 退出规则

状态原因模式匹配

onStatusReason 参数最多支持 512 个字符的模式匹配。模式可以使用通配符 (*)，并与 SageMaker AI 返回的状态原因进行匹配。

对于服务作业， SageMaker 来自 AI 的状态消息以 “接收的状态来自 SageMaker:” 为前缀，以将其与 AWS Batch生成的消息区分开来。常见模式包括：

- `Received status from SageMaker: InternalServerError*`：匹配内部服务错误
- `Received status from SageMaker: ValidationException*`：匹配客户端验证错误
- `Received status from SageMaker: ResourceLimitExceeded*`：匹配资源限制错误

- `*CapacityError*`：匹配与容量相关的失败

> **Tip**
>
> 使用特定的模式匹配来适当地恰当处理不同的错误类型。例如，重试内部服务器错误，但如果出现表明作业参数存在问题的验证错误，则立即失败。

监控 AWS Batch 队列中的服务作业

您可以使用`list-service-jobs`、和监控 SageMaker 训练作业队列中作业的状态`get-job-queue-snapshot`。

查看队列中正在运行的作业：

```
aws batch list-service-jobs \
  --job-queue my-sm-training-fifo-jq \
  --job-status RUNNING
```

查看队列中正在等待的作业：

```
aws batch list-service-jobs \
  --job-queue my-sm-training-fifo-jq \
  --job-status RUNNABLE
```

查看已提交 SageMaker 但尚未运行的作业：

```
aws batch list-service-jobs \
  --job-queue my-sm-training-fifo-jq \
  --job-status SCHEDULED
```

获取队列中排名最前的作业快照：

```
aws batch get-job-queue-snapshot --job-queue my-sm-training-fifo-jq
```

此命令会显示队列中即将执行的服务作业的顺序。

获取详细的服务作业信息

使用 DescribeServiceJob 操作可以获取有关特定服务作业的全面信息，包括其当前状态、服务资源标识符和详细尝试信息。

查看有关特定作业的详细信息：

```
aws batch describe-service-job \
  --job-id a4d6c728-8ee8-4c65-8e2a-9a5e8f4b7c3d
```

此命令会返回有关此作业的全面信息，包括：

- 作业 ARN 和当前状态
- 服务资源标识符（例如 SageMaker 训练作业 ARN）
- 调度优先级和重试配置
- 包含原始服务参数的服务请求有效载荷
- 详细尝试信息，包括启动和停止时间
- 来自目标服务的状态消息

监控 SageMaker 培训作业

通过监控 SageMaker 训练作业时 AWS Batch，您可以访问 AWS Batch 作业信息和基础 SageMaker 培训作业详细信息。

任务详情中的服务资源标识符包含 SageMaker 训练作业 ARN：

```
{
  "latestAttempt": {
    "serviceResourceId": {
      "name": "TrainingJobArn",
      "value": "arn:aws:sagemaker:us-east-1:123456789012:training-job/my-training-job"
    }
  }
}
```

您可以使用此 ARN 直接从以下地址获取更多详情： SageMaker

```
aws sagemaker describe-training-job \
```

```
  --training-job-name my-training-job
```

通过检查 AWS Batch 状态和 SageMaker 训练作业状态来监控作业进度。 AWS Batch 作业状态显示整个作业生命周期，而 SageMaker 训练作业状态则提供有关训练过程的特定于服务的详细信息。

终止服务作业

使用 TerminateServiceJob 操作停止正在运行的服务作业。

终止特定的服务作业：

```
aws batch terminate-service-job \
  --job-id a4d6c728-8ee8-4c65-8e2a-9a5e8f4b7c3d \
  --reason "Job terminated by user request"
```

当您终止服务作业时，会 AWS Batch 停止该任务并通知目标服务。对于 SageMaker 训练作业，这也将停止 SageMaker AI 中的训练作业。

任务状态

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。

> **Note**
>
> 阵列作业父项PENDING会更新到任何子作业更新到的时间，RUNNABLE并在子作业运行时保持其PENDING状态。要查看这些作业，请按PENDING状态筛选，直到所有子作业都达到终止状态。

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 完成持续时间和 Amazon ECS containerDependency 解析持续时间处于“启动”状态。为作业提取映像所用时间等于作业处于 STARTING 状态的时长。

例如，如果提取作业的映像需要三分钟，则您的作业将处于“启动”状态三分钟。如果 initContainers 总共需要十分钟才能完成，则您的 Amazon EKS 作业将在“启动”状态下持续十分钟。如果您的 Amazon ECS 作业中设置了 Amazon ECS containerDependencies，则在解析所有容器依赖项（其运行时）之前，该作业将处于“启动”状态。“启动”不包含在超时中；持续时间从“运行”开始。有关更多信息，请参阅作业状态。

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 Lo CloudWatch gs 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 Lo CloudWatch gs 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 Lo CloudWatch gs 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 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

此变量设置为运行容器组（pod）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。此变量不存在于主节点上，但设置为作业主节点的私有 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evaluateOnExit已指定，则它最多可以包含 5 个重试策略。如果evaluateOnExit已指定，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 任务状态。

> **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、教程：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 IDs 该作业所依赖的作业。当你这样做时， AWS Batch 调度器会确保你的作业只有在指定的依赖关系成功完成后才会运行。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，请参阅教程：提交作业。

资源感知调度 允许您根据运行作业所需的消耗性资源来调度作业。您可以指定作业运行所需的消耗性资源，Batch 在调度作业时会考虑这些资源依赖项。您可以通过仅分配具有全部所需资源的作业来减少计算资源利用不足的情况。资源感知调度支持 FIFO 和公平份额调度策略，并且可以与 Batch 支持的所有计算平台结合使用，包括 EKS、ECS 和 Fargate。此功能可用于数组作业、多节点并行（MNP）作业和常规 Batch 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，请参阅教程：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，该eksPropertiesOverride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。

主题

教程：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。

教程：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
- 使用公平份额调度来帮助调度作业

Kubernetes`Secrets` 和 `ServiceAccounts`

AWS Batch 支持引用Kubernetes`Secrets`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

- 运行 GPU 作业
- 作业在RUNNABLE状态卡住

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（也称为分组调度），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 AWS Batch DescribeJobsAPI 操作描述 *aws_batch_job_id* #1。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。

主题

- 环境变量
- 节点组
- MNP 作业的作业生命周期
- MNP 的计算环境注意事项 AWS Batch

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。

MNP 作业的作业生命周期

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。主节点是一个子任务， AWS Batch 用于监控以确定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。

MNP 的计算环境注意事项 AWS Batch

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。
- 与计算环境关联的安全组不得超过 5 个。创建并附加到 MNP 任务的弹性网络接口将使用在计算环境中指定的安全组。如果未指定安全组，则会使用 VPC 的默认安全组。

- 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。
- 您的计算环境必须有足够的最大 v CPUs 才能支持您的多节点 parallel 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。

Amazon EKS 上的多节点并行作业

您可以 AWS Batch 在 Amazon Elastic Kubernetes Service 上使用在托管集群上运行多节点并行 (MNP) 任务（也称为群组调度）。Kubernetes此选项通常用于无法在单个 Amazon Elastic Compute Cloud 实例上运行的大型的紧密耦合高性能作业。有关更多信息，请参阅 多节点并行作业。

您可以使用此功能运行 Amazon EKS 托管的特定于 Kubernetes 的高性能计算应用程序、大型语言模型训练和其他人工智能（AI）/机器学习（ML）作业。

主题

- 运行 MNP 作业
- 创建 Amazon EKS MNP 作业定义
- 提交 Amazon EKS MNP 作业
- 覆盖 Amazon EKS MNP 作业定义

运行 MNP 作业

AWS Batch 支持使用亚马逊 EC2 的亚马逊弹性容器服务和亚马逊 EKS 上的 MNP 任务。以下内容提供有关该功能的实例和容器参数的更多细节。

Amazon EKS 上的 MNP 的实例配额

- 单个 MNP 作业最多可使用 1000 个实例。

- 最多可以有 5000 个实例加入单个 Amazon EKS 集群。
- 最多可以将 5 个计算环境集群化并附加到作业队列。

例如，您可以在一个作业队列中最多扩展 5 个集群计算环境，在每个计算环境中最多可扩展 1000 个实例。

除了实例参数外，还需要注意的是，您不能通过任何一种服务使用 Fargate 进行 MNP 作业。

您只能在每个 MNP 作业中使用一种实例类型。您可以通过更新计算环境，或者在定义新的计算环境时更改实例类型。您还可以指定实例类型，并在创建作业定义时提供 vCPU 和内存要求。

Amazon EKS 上的 MNP 的容器配额

- 多节点并行作业为每个节点支持一个容器组（pod）。
- 每个容器组（pod）最多 10 个容器（或 10 个 init 容器。有关更多信息，请参阅 Kubernetes 文档中的 Init 容器。）。
- 每个 MNP 作业中最多 5 个节点范围。
- 每个节点范围内最多 10 个不同的容器映像。

例如，在包含 5 个节点范围和总共 50 个唯一映像的单个 MNP 作业中，您最多可以运行 10000 个容器。

在私有 Amazon VPC 和 Amazon EKS 集群中运行 MNP 作业

MNP 作业可以在任何 Amazon EKS 集群上运行，无论该集群是否有公共互联网。使用只能访问私有网络的 Amazon EKS 集群时，请确保该集群 AWS Batch 可以访问 Amazon EKS 控制平面和托管 Kubernetes API 服务器。您可以通过 Amazon Virtual Private Cloud 端点授予必要的访问权限。有关更多信息，请参阅 Configure an endpoint service。

Amazon EKS 集群容器组（pod）无法从公共来源下载映像，因为私有 VPC 无法访问互联网。您的 Amazon EKS 集群必须从 Amazon VPC 中的容器注册表中拉取映像。您可以在 Amazon VPC 中创建Amazon Elastic Container Registry（Amazon ECR），并将容器映像复制到其中，以供节点访问。

您还可以使用 Amazon ECR 创建缓存提取规则。为外部公有注册表创建缓存提取规则后，您就可以使用 Amazon ECR 私有注册表 URI 从该外部公有注册表中提取映像。然后，Amazon ECR 会创建一个存储库并缓存映像。当使用 Amazon ECR 私有注册表 URI 提取缓存镜像时，Amazon ECR 会检查远程注册表以查看是否有新版本的镜像，并且会最多每 24 小时更新一次您的私有注册表。有关更多信息，请参阅 Creating a pull through cache rule in Amazon ECR。

错误通知

如果您的 MNP 职位被屏蔽，您可以通过 AWS 管理控制台 和 Amazon EventBridge 收到通知。例如，如果 MNP 作业卡在队列的开头，您会收到有关该问题的通知以及导致该问题的原因的信息，以便您可以立即采取措施来解除对作业队列的阻止。您还可以选择在一段特定的时间内没有执行任何操作后自动终止 MNP 作业，具体时长可以在作业队列模板中定义。有关更多信息，请参阅 作业队列阻塞事件。

创建 Amazon EKS MNP 作业定义

要在 Amazon EKS 上定义和运行 MNP 作业，RegisterJobDefinition 和 SubmitJob API 操作中有新的参数。

- 在 nodeProperties 部分下使用 eksProperties 来定义您的 MNP 作业定义。
- 在提交 MNP 作业时，在 nodePropertyOverrides 部分下使用 eksPropertiesOverride 来覆盖作业定义中定义的参数。

这些操作可以通过 API 操作和 AWS 管理控制台来定义。

参考：注册 Amazon EKS MNP 作业定义请求有效载荷

以下示例说明了如何使用两个节点注册 Amazon EKS MNP 作业定义。

```
{
  "jobDefinitionName": "MyEksMnpJobDefinition",
  "type": "multinode",
  "nodeProperties": {
    "numNodes": 2,
    "mainNode": 0,
    "nodeRangeProperties": [
      {
        "targetNodes" : "0:",
        "eksProperties": {
          "podProperties": {
            "containers": [
              {
                "name": "test-eks-container-1",
                "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                "command": [
                  "sleep",
                  "60"
                ],
```

```
                "resources": {
                  "limits": {
                    "cpu": "1",
                    "memory": "1024Mi"
                  }
                },
                "securityContext":{
                  "runAsUser":1000,
                  "runAsGroup":3000,
                  "privileged":true,
                  "readOnlyRootFilesystem":true,
                  "runAsNonRoot":true
               }
              }
            ],
            "initContainers": [
               {
                  "name":"init-ekscontainer",
                  "image": "public.ecr.aws/amazonlinux/amazonlinux:2",
                  "command": [
                     "echo",
                     "helloWorld"
                   ],
                   "resources": {
                     "limits": {
                       "cpu": "1",
                       "memory": "1024Mi"
                     }
                  }
               }
            ],
            "metadata": {
               "labels": {
                  "environment" : "test"
               }
            }
          }
        }
      }
    ]
  }
}
```

要使用注册作业定义 AWS CLI，请将定义复制到名为 MyEksMnpJobDefinition.json 的本地文件中，然后运行以下命令。

```
aws batch register-job-definition --cli-input-json file://MyEksMnpJobDefinition.json
```

您将收到以下 JSON 响应。

```
{
    "jobDefinitionName": "MyEksMnpJobDefinition",
    "jobDefinitionArn": "arn:aws:batch:us-east-1:0123456789:job-definition/
MyEksMnpJobDefinition:1",
    "revision": 1
}
```

提交 Amazon EKS MNP 作业

要使用已注册的作业定义提交作业，请输入以下命令。将 <EKS_JOB_QUEUE_NAME> 的值替换为与 Amazon EKS 计算环境关联的预先存在的作业队列的名称或 ARN。

```
aws batch submit-job --job-queue <EKS_JOB_QUEUE_NAME> \
    --job-definition MyEksMnpJobDefinition \
    --job-name myFirstEksMnpJob
```

您将收到以下 JSON 响应。

```
{
    "jobArn": "arn:aws:batch:region:account:job/9b979cce-9da0-446d-90e2-ffa16d52af68",
    "jobName": "myFirstEksMnpJob",
    "jobId": "<JOB_ID>"
}
```

您可以使用以下命令通过返回的 jobId 检查作业的状态。

```
aws batch describe-jobs --jobs <JOB_ID>
```

覆盖 Amazon EKS MNP 作业定义

您可以选择覆盖作业定义详细信息（例如更改 MNP 作业规模或子作业详细信息）。以下提供了提交五节点 MNP 作业的 JSON 请求有效载荷示例，以及对 `test-eks-container-1` 容器命令的更改。

```
{
  "numNodes": 5,
  "nodePropertyOverrides": [
    {
      "targetNodes": "0:",
      "eksPropertiesOverride": {
        "podProperties": {
          "containers": [
            {
              "name": "test-eks-container-1",
              "command": [
                "sleep",
                "150"
              ]
            }
          ]
        }
      }
    }
  ]
}
```

要提交包含这些覆盖的作业，请将示例保存到本地文件 eks-mnp-job-nodeoverride.json 中，然后使用 AWS CLI 提交带有替代项的作业。

数组作业

数组作业是一种共享公共参数的作业，例如作业定义CPUs、v 和内存。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。

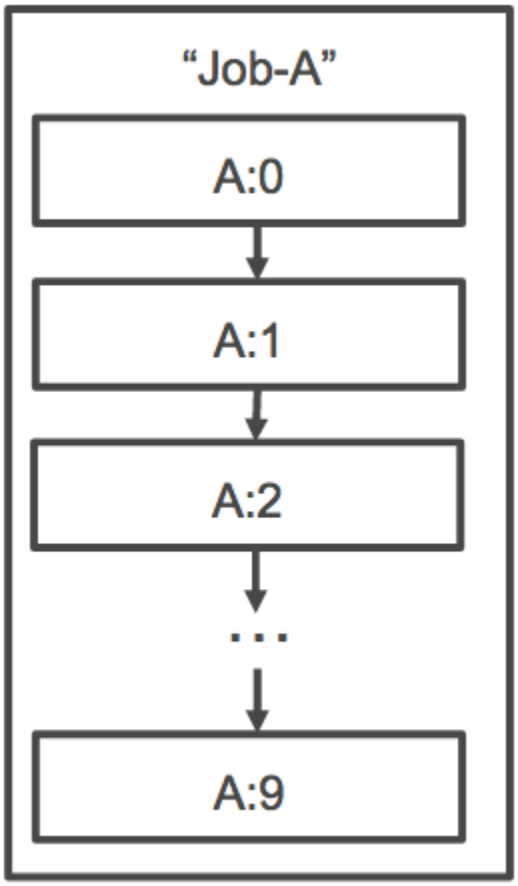
AWS Batch 阵列作业的提交方式与普通作业一样。但是，您可以指定一个数组大小 (介于 2 和 10000 之间) 来定义该数组中应运行的子作业数。如果您提交一个数组大小为 1000 的作业，则单个作业会运行并生成 1000 个子作业。该数组作业是用于管理所有子作业的参考或指针。这种方式将允许您使用单个查询提交大型工作负载。`attemptDurationSeconds` 参数中指定的超时适用于每个子作业。父阵列作业没有超时。

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`。

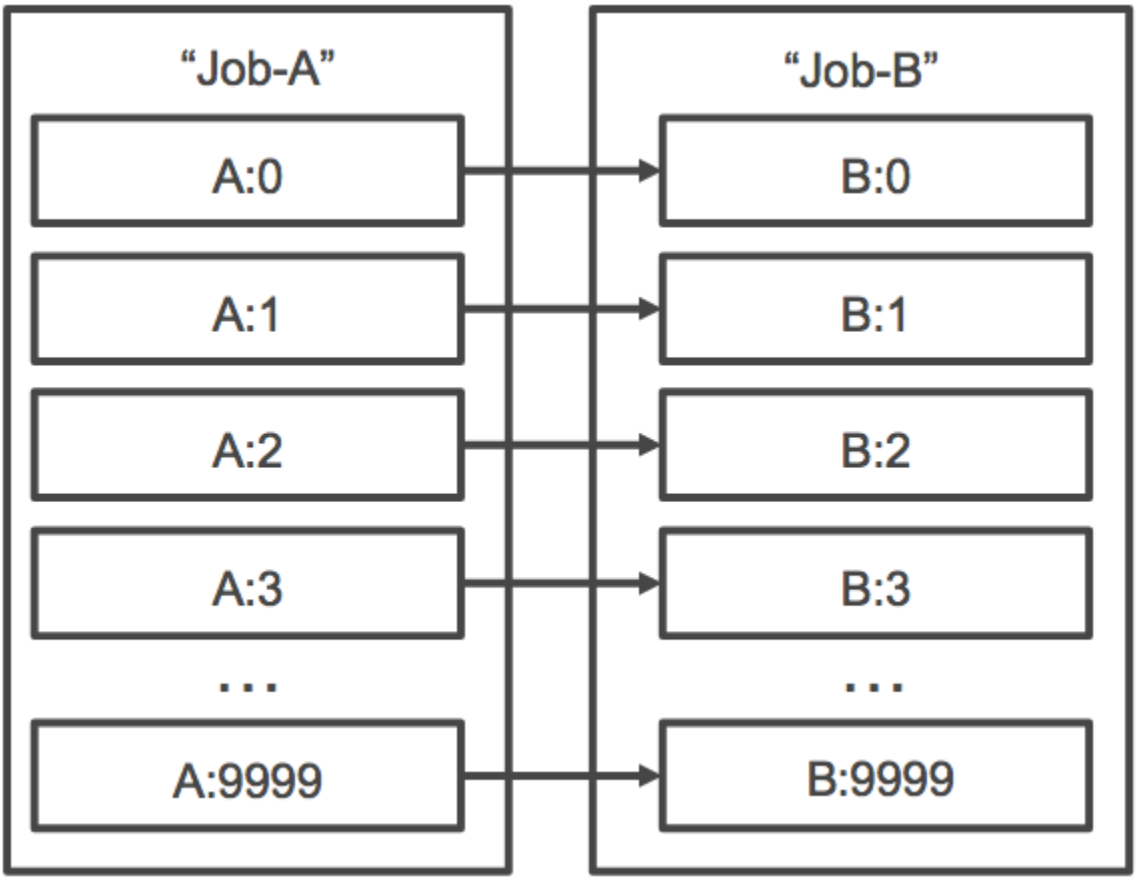
父阵列作业可以输入 SUBMITTED、PENDING、FAILED 或 SUCCEEDED 状态。当任何子作业更新为 RUNNABLE时，阵列父作业将更新为 PENDING。有关作业依赖关系的更多信息，请参阅 作业依赖项。

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 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作业），则父作业也会失败。在这种情况下，父作业会在所有子作业完成时转换为 FAILED。

有关搜索和筛选阵列作业的更多信息，请参阅搜索作业队列中的作业。

主题

- 数组作业工作流示例
- 使用数组作业索引控制作业差异化

数组作业工作流示例

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://amzn-s3-demo-source-bucket",
        "Recipient": "pipeline-notifications@mycompany.com"
    },
    "dependsOn": [
        {
            "jobId": "JobD_job_ID"
        }
    ]
}
```

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。

主题

先决条件

本教程工作流包含以下先决条件：

- 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。

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.
```

将映像推送到 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 管理控制台。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
```

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 管理控制台。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
```

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。

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。

运行 GPU 作业

GPU 作业可帮助您运行使用实例的作业 GPUs。

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 G6 实例、Amazon EC2 P2 实例、Amazon EC2 P3 实例、Amazon EC2 P4d 实例、Amazon EC2 P5 实例、Amazon EC2 P6 实例、Amazon EC2 Trn1 实例、Amazon EC2 Trn2 实例、Amazon EC2 Inf1 实例、Amazon EC2 Inf2 实例、Amazon EC2 DI1 实例 和 Amazon EC2 DI2 实例。

实例类型	GPUs	GPU 内存	v CPUs	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

实例类型	GPUs	GPU 内存	v CPUs	内存	网络带宽
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
g5g.xlarge	1	16 GiB	4	8 GiB	最高 10 Gbps
g5g.2xlarge	1	16 GiB	8	16 GiB	最高 10 Gbps
g5g.4xlarge	1	16 GiB	16	32 GiB	最高 10 Gbps
g5g.8xlarge	1	16 GiB	32	64 GiB	12 Gbps
g5g.16xlarge	2	32 GiB	64	128 GiB	25 Gbps
g5g.metal	2	32 GiB	64	128 GiB	25 Gbps
g6.xlarge	1	24 GiB	4	16 GiB	最高 10 Gbps
g6.2xlarge	1	24 GiB	8	32 GiB	最高 10 Gbps
g6.4xlarge	1	24 GiB	16	64 GiB	最高 25 Gbps
g6.8xlarge	1	24 GiB	32	128 GiB	25 Gbps

实例类型	GPUs	GPU 内存	v CPUs	内存	网络带宽
g6.16xlarge	1	24 GiB	64	256 GiB	25 Gbps
g6.12xlarge	4	96 GiB	48	192 GiB	40Gbps
g6.24xlarge	4	96 GiB	96	384 GiB	50 Gbps
g6.48xlarge	8	192 GiB	192	768 GiB	100 Gbps
g6e.xlarge	1	48 GiB	4	32 GiB	高达 20 Gbps
g6e.2xlarge	1	48 GiB	8	64 GiB	高达 20 Gbps
g6e.4xlarge	1	48 GiB	16	128 GiB	20 Gbps
g6e.8xlarge	1	48 GiB	32	256 GiB	25 Gbps
g6e.16xlarge	1	48 GiB	64	512 GiB	35 Gbps
g6e.12xlarge	4	192 GiB	48	384 GiB	100 Gbps
g6e.24xlarge	4	192 GiB	96	768 GiB	200 Gbps
g6e.48xlarge	8	384 GiB	192	1536 GiB	400 Gbps
gr6.4xlarge	1	24 GiB	16	128 GiB	最高 25 Gbps
gr6.8xlarge	1	24 GiB	32	256 GiB	25 Gbps
p2.xlarge	1	12 GiB	4	61 GiB	高
p2.8xlarge	8	96 GiB	32	488 GiB	10 Gbps
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

实例类型	GPUs	GPU 内存	v CPUs	内存	网络带宽
p3dn.24xlarge	8	256 GiB	96	768 GiB	100 Gbps
p4d.24xlarge	8	320 GiB	96	1152 GiB	400 Gbps
p4de.24xlarge	8	640 GiB	96	1152 GiB	400 Gbps
p5.48xlarge	8	640 GiB	192	2 TiB	3200 Gbps
p5e.48xlarge	8	1128 GiB	192	2 TiB	3200 Gbps
p5en.48xlarge	8	1128 GiB	192	2 TiB	3200 Gbps
p6-b200.48xlarge	8	1440 GiB	192	2 TiB	100 Gbps
trn1.2xlarge	1	32 GiB	8	32 GiB	最高 12.5Gbps
trn1.32xlarge	16	512 GiB	128	512 GiB	800 Gbps
trn1n.32xlarge	16	512 GiB	128	512 GiB	1600 Gbps
trn2.48xlarge	16	1.5 TiB	192	2 TiB	3.2 Tbps
inf1.xlarge	1	8 GiB	4	8 GiB	最高 25 Gbps
inf1.2xlarge	1	8 GiB	8	16 GiB	最高 25 Gbps
inf1.6xlarge	4	32 GiB	24	48 GiB	25 Gbps
inf1.24xlarge	16	128 GiB	96	192 GiB	100 Gbps
inf2.xlarge	1	32 GiB	4	16 GiB	最高 15 Gbps
inf2.8xlarge	1	32 GiB	32	128 GiB	最高 25 Gbps
inf2.24xlarge	6	192 GiB	96	384 GiB	50 Gbps
inf2.48xlarge	12	384 GiB	192	768 GiB	100 Gbps
dl1.24xlarge	8	256 GiB	96	768 GiB	400 Gbps

实例类型	GPUs	GPU 内存	v CPUs	内存	网络带宽
dl2q.24xlarge	8	128 GiB	96	768 GiB	100 Gbps

> **Note**
>
> 对于 GPU 任务， AWS Batch 仅支持采用 NVIDIA 的实例类型 GPUs。例如，G4ad 系列不支持 GPU 调度。您仍然可以在G4ad上使用， AWS Batch 方法是在任务定义中仅定义 vCPU 和内存要求，然后使用亚马逊 ECS 或 Amazon EKS 计算优化的 AMI，或者使用自定义 AMD 的自定义 AMI，通过在 Amazon EC2 启动模板用户数据中进行自定义，直接访问主机 GPU。
> GPUs
> 自定义 AMIs提供给的 GPU 任务支持使用 ARM64 架构的实例类型， AWS Batch 或者 GPUs 通过自定义代码和配置访问 Amazon EC2 用户数据。例如，G5g 实例系列。

任务定义的 resource Requirements 参数指定 GPUs 要固定到容器的数量。在作业持续时间内，在该实例上运行的任何其他作业都 GPUs 无法使用此数字。计算环境中将运行 GPU 作业的所有实例类型都必须来自 p6、p3、p4、p5、g3、g3s、g4、g5 或 g6 实例系列。如果不这么做，GPU 作业可能会停滞于 RUNNABLE 状态。

不使用的作业 GPUs 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。

主题

- 在 Amazon EKS 上创建基于 GPU 的 Kubernetes 集群
- 创建 Amazon EKS GPU 作业定义
- 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 开始使用 Amazon EKS 上的 AWS Batch 中的步骤。此外，请考虑以下操作：

- AWS Batch 支持 NVIDIA 的实例类型 GPUs。

- 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 作业定义

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 GPUs中的日程安排。

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"
}
```

查看 AWS Batch 作业队列中的作业

您可以在 AWS Batch中查看和筛选作业。该功能提供了一个用来查看现有作业队列以及按三个选项之一筛选队列中作业的选项。

使用“搜索和筛选”功能可以检索非处于最终状态（SUCCEEDED 或 FAILED）的作业。一旦作业的状态变为 SUCCEEDED 或 FAILED，您最多可以在七天内检索该作业。您仍然可以查看任务 CloudWatch 或 Amazon EventBridge 日志。

使用此过程可在 AWS Batch 控制台中列出作业队列中的所有作业。也可使用筛选结果字段来根据您指定的条件缩小结果范围。

1. 导航至 AWS Batch 控制台。
2. 在导航窗格中，选择作业。
3. 展开作业队列下拉列表，选择要在其中进行搜索的作业队列。

   > **Note**
   >
   > 您一次只能在一个作业队列中搜索作业。

4. 在筛选结果字段中，输入要包含在结果中的关键字。您可以使用此字段按任务名称、状态或作业 ID 进行筛选。根据属性的不同，可能还必须定义其他运算符，例如等于（=）或包含（:）。

   > **Note**
   >
   > SageMaker 训练作业队列仅支持按任务名称和作业 ID 筛选

5. 选择搜索。

在 AWS Batch 作业队列中搜索作业

你可以使用 Job Search 搜索和 AWS Batch 筛选职位。此功能提供了搜索现有作业队列以及筛选队列中作业的选项。

使用“搜索和筛选”功能可以检索非处于最终状态（SUCCEEDED 或 FAILED）的作业。一旦作业的状态变为 SUCCEEDED 或 FAILED，您最多可以在七天内检索该作业。您仍然可以查看任务 CloudWatch 或 Amazon EventBridge 日志。

要同时使用多个条件进行搜索，请使用高级搜索功能。例如，您可以使用以下任何或所有筛选条件：状态、日期范围和其他条件（例如作业名称、作业定义，或作业 ID）。

搜索 AWS Batch 职位（AWS 控制台）

使用此过程可在 AWS Batch 控制台中搜索作业队列中的作业。

1. 导航至 AWS Batch 控制台。
2. 在导航窗格中，选择作业。
3. 打开高级搜索。
4. 展开作业队列下拉列表，选择要在其中进行搜索的作业队列。

> **Note**
>
> 您一次只能在一个作业队列中搜索作业。

5. 对于搜索选项：

   a. 在状态下拉列表中，您可以选择一个或多个状态进行筛选。有关更多信息，请参阅任务状态和服务作业状态。

   > **Note**
   >
   > 阵列作业父项PENDING会更新到任何子作业更新到的时间，RUNNABLE并在子作业运行时保持其PENDING状态。要查看这些作业，请按PENDING状态筛选，直到所有子作业都达到终止状态。

   b. 选择日期范围，以根据日期和时间范围筛选结果。

   - 选择相对模式，以搜索创建日期在当前日期和时间起倒数的时间范围内的作业。
   - 选择绝对模式，以搜索创建日期在指定日期和时间范围内的作业。

   c. 在其他条件字段中，输入要包含在搜索结果中的关键字。例如，您可以使用此字段按作业名称、作业定义、作业 ID 或共享标识符进行搜索。根据属性的不同，可能还必须定义其他运算符，例如等于（=）或包含（:）。

   > **Note**
   >
   > SageMaker 训练作业队列仅支持按任务名称和作业 ID 筛选

> **Note**
>
> 按共享标识符筛选时，您还可以指定任务状态。这是限制的例外，其他过滤器不包括作业状态筛选。

6. 选择搜索。

搜索和筛选 AWS Batch 职位 (AWS CLI)

使用此过程通过 AWS CLI列出作业队列中的所有作业。也可使用 -filters 参数来根据您指定的条件缩小结果范围。

Search job queue (AWS CLI)

您可以使用 list-jobs 命令来搜索和筛选作业队列。

例如，您可以按作业名称搜索作业队列：

```
aws batch list-jobs \
    --job-queue my-job-queue \
    --filters name=JOB_NAME,values="my-job"
```

按共享标识符筛选作业：

```
aws batch list-jobs \
    --job-queue my-job-queue \
    --filters name=SHARE_IDENTIFIER,values="my-share"
```

按共享标识符筛选时，可以包括任务状态：

```
aws batch list-jobs \
    --job-queue my-job-queue \
    --job-status RUNNING \
    --filters name=SHARE_IDENTIFIER,values="my-share"
```

在前面的命令中，进行以下更改：

- *my-job-queue*替换为任务队列的名称。

- *my-job*用您的任务名称替换。
- *my-share*替换为要筛选依据的股票标识符。

Search service job queue (AWS CLI)

您可以使用list-service-jobs命令来搜索和筛选服务作业队列。

例如，您可以按作业名称搜索服务作业队列：

```
aws batch list-service-jobs \
    --job-queue my-sm-queue \
    --filters name=JOB_NAME,values="my-sm-job"
```

按共享标识符筛选服务作业：

```
aws batch list-service-jobs \
    --job-queue my-sm-queue \
    --filters name=SHARE_IDENTIFIER,values="my-share"
```

在前面的命令中，进行以下更改：

- *my-sm-queue*替换为服务任务队列的名称。
- *my-sm-job*替换为服务任务的名称。
- *my-share*替换为要筛选依据的股票标识符。

AWS Batch 作业的联网模式

下表描述了 AWS Batch 作业类型的联网模式和典型用法。有关注意事项和行为的更多详细信息，请参见“作业类型”列中的链接。

作业类型	支持的网络模式	典型用法
ECS-EC2 简单作业	host	用于可扩展性最高且高度并行的批处理工作负载，这些工作负载只需要输出到计算环境中所定义 vpc。

作业类型	支持的网络模式	典型用法
ECS-EC2 多节点并行作业	awsvpc	用于采用单作业模式的紧密耦合型多主机（节点）分布式工作负载，并在任务节点之间协调通信。
ECS-Fargate 简单作业	awsvpc	真正的无服务器，用于高度并行的批处理工作负载。通常为 TCO 最低且容器隔离程度最高的作业模式。
EKS-EC2 简单作业	主机和容器组	用于高度可扩展且高度并行的批处理工作负载，这些工作负载只需要输出到计算环境中所定义 vpc。默认为主机网络。
EKS-EC2 多节点并行作业	主机和容器组	用于采用单作业模式的紧密耦合型多主机（节点）分布式工作负载，并在容器组（pod）节点之间协调通信。默认为主机网络。

在“日志”中查看 AWS Batch 作业 CloudWatch 日志

您可以将 AWS Batch 任务配置为向 Amazon Logs 发送 CloudWatch 日志信息。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)。有关作业队列中作业排序和筛选的更多详细信息，请参阅查看 AWS Batch 作业队列中的作业和搜索作业队列中的作业
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 在 AWS Batch 作业队列中搜索作业。

4. 在状态 中，选择所需的作业状态。
5. 选择需要的作业，这时将会打开详细信息页面。
6. 在详细信息页面上，向下滚动到日志流名称并选择该链接。该链接将打开该任务的 Amazon CloudWatch 日志页面。
7. （可选）如果这是您首次查看日志，则可能会要求您授权。

   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。

查看 AWS Batch 工作信息

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 在 AWS Batch 作业队列中搜索作业。

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。

安全性 AWS Batch

云安全 AWS 是重中之重。作为 AWS 客户，您可以从专为满足最安全敏感的组织的要求而构建的数据中心和网络架构中受益。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 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

Identity and Access Management 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

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅排查 AWS Batch IAM）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅如何 AWS Batch 与 IAM 配合使用）
- IAM 管理员：编写用于管理访问权限的策略（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 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center （例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，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，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，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

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

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。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管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。示例包括 IAM 角色信任策略和 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。您必须在基于资源的策略中指定主体。

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。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在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。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
ACLs	否
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	是
服务关联角色	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》中的使用客户管理型策略定义自定义 IAM 权限。

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。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。

AWS Batch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。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。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 AWS Batch 资源类型及其列表 ARNs，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。

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。

`Condition` 元素根据定义的条件指定语句何时执行。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。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。

基于属性的访问控制 (ABAC) AWS Batch

支持 ABAC（策略中的标签）：是

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

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》中的使用 ABAC 授权定义权限。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）。

将临时证书用于 AWS Batch

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。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 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的AWS 服务。

AWS Batch的跨服务主体权限

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。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

支持服务关联角色：是

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关联角色的权限。

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。

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。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。

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（控制台）。

有关由 AWS Batch定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》 AWS Batch中的操作、资源和条件密钥。 ARNs

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，请遵循以下指南和建议：

- 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 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》中的使用 IAM Access Analyzer 验证策略。

- 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》中的使用 MFA 保护 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 中的安全最佳实践。

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": "*"
        }
    ]
}
```

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 将服务关联角色用于 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。

要查看策略的 JSON，请参阅AWS 托管策略参考指南BatchServiceRolePolicy中的。

AWS 托管策略：AWSBatchServiceRolePolicyForSageMaker

AWSServiceRoleForAWSBatchWithSagemaker AWS Batch 允许代表您执行操作。您不能将此策略附加到您的 IAM 实体。有关更多信息，请参阅 将服务关联角色用于 AWS Batch。

此策略 AWS Batch 允许对特定资源完成以下操作：

- sagemaker— AWS Batch 允许管理 SageMaker AI 训练作业和其他 SageMaker AI 资源。
- iam:PassRole— AWS Batch 允许将客户定义的执行角色传递给 SageMaker AI 以执行作业。资源限制允许将角色传递给 SageMaker AI 服务。

要查看策略的 JSON，请参阅AWS 托管策略参考指南AWSBatchServiceRolePolicyForSageMaker中的。

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。遵循授予最低权限的标准安全建议，AWSBatchServiceRole 托管策略可用作指南。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，但最好使用与服务相关的角色，这样可以减少出错率、扩大范围并改善托管体验。

- `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。

要查看策略的 JSON，请参阅AWS 托管策略参考指南AWSBatchServiceRole中的。

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。

要查看策略的 JSON，请参阅AWS 托管策略参考指南AWSBatchFullAccess中的。

AWS Batch AWS 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
AWSBatchServiceRolePolicyForSageMaker策略已添加	为 AWSBatchServiceRolePolicyForSageMaker服务相关角色添加了新的 AWS 托管策略，AWS Batch 允许代表您管理 SageMaker AI。	2025 年 7 月 31 日
BatchServiceRolePolicy政策已更新	进行了更新，以增加对描述竞价型实例集请求历史记录和 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 IDs、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
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	已更新，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

更改	描述	日期
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 已开始跟踪更改	AWS Batch 开始跟踪其 AWS 托管策略的更改。	2021 年 3 月 10 日

AWS Batch IAM 策略、角色和权限

默认情况下，用户无权创建或修改 AWS Batch 资源或使用 AWS Batch API、 AWS Batch 控制台或执行任务 AWS CLI。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 代表你给他 AWS 服务 人打电话。因此， AWS Batch 必须使用您的凭据进行身份验证。更具体地说，通过创建提供这些权限的 IAM 角色和策略 AWS Batch 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Amazon ECS 实例角色，请参阅 IAM 用户指南中的 IAM 角色、使用服务相关角色和创建角色向 AWS 服务委派权限。

主题

- IAM 策略结构
- 资源：策略示例 AWS Batch
- 资源： AWS Batch 托管策略

IAM 策略结构

以下主题说明 IAM policy 的结构。

主题

策略语法

IAM 策略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"
    }
      }
    }
  ]
}
```

组成语句的四个主要元素如下：

- 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 适用于 API 的操作 AWS Batch。
- 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API 操作支持的资源级权限 AWS Batch和的 Amazon 资源名称 AWS Batch。如果 AWS Batch API 操作当前不支持资源级权限，请添加通配符 (*) 以指定所有资源都可能受到该操作的影响。

- 条件：条件是可选的。它们可以用于控制策略生效的时间。

有关的 IAM 策略声明的示例 AWS Batch，请参阅资源：策略示例 AWS Batch。

适用于 API 的操作 AWS Batch

在 IAM 策略语句中，您可以从支持 IAM 的任何服务中指定任何 API 操作。对于 AWS Batch，请使用以下前缀和 API 操作的名称：`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，请添加通配符 (*)。

```
"Action": "batch:*"
```

有关 AWS Batch 操作列表，请参阅 AWS Batch API 参考中的操作。

的 Amazon 资源名称 AWS Batch

每个 IAM 政策声明都适用于您使用其 Amazon 资源名称 (ARNs) 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`)。

region

代表 AWS 区域 资源（例如，`us-east-2`）。

account

不带连字符的 AWS 账户 ID（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 操作目前支持对多个 API 操作的资源级权限。有关更多信息，请参阅 API 操作支持的资源级权限 AWS Batch。要指定所有资源，或者如果特定 API 操作不支持 ARNs，请在Resource元素中添加通配符 (*)。

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

确认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DecodeAuthorizationMessage AWS Security Token ServiceAPI 参考和AWS CLI 命令参考decode-authorization-message中的。

资源：策略示例 AWS Batch

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。

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

- 资源：的只读访问权限 AWS Batch
- 资源：限制为 POSIX 用户、Docker 映像、权限级别和作业提交角色
- 资源：限于作业提交的作业定义前缀
- 资源：限于作业队列
- 当所有条件都匹配字符串时拒绝操作
- 资源：当所有条件键都匹配字符串时拒绝操作
- 资源：使用 batch:ShareIdentifier 条件键
- 使用管理 SageMaker AI 资源 AWS Batch
- 资源：通过作业定义和作业队列上的资源标签限制作业提交

资源：的只读访问权限 AWS Batch

以下策略授予用户使用名称以`Describe`和开头的所有 AWS Batch API 操作的权限`List`。

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。

JSON

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

```
            "Action": [
                "batch:Describe*",
                "batch:List*",
                "batch:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

资源：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。

使用第一和第二条语句注册和取消注册名称前缀为的任何作业定义名称。*JobDefA_*

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 Reference》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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:RegisterJobDefinition"
```

```
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:999999999999:job-definition/JobDefA_*"
            ],
            "Condition": {
                "StringEquals": {
                    "batch:User": [
                        "nobody"
                    ],
                    "batch:Image": [
                        "999999999999.dkr.ecr.us-east-2.amazonaws.com/myImage"
                    ]
                },
                "Bool": {
                    "batch:Privileged": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "batch:DeregisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:999999999999:job-definition/JobDefA_*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::999999999999:role/MyBatchJobRole"
            ]
        }
    ]
}
```

资源：限于作业提交的作业定义前缀

使用以下策略向任何作业定义名称以开头的作业队列提交作业*JobDefA*。

> **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。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:111122223333:job-definition/JobDefA_*",
                "arn:aws:batch:us-east-2:111122223333:job-queue/*"
            ]
        }
    ]
}
```

资源：限于作业队列

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。

JSON

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

```
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": [
                "arn:aws:batch:us-east-2:888888888888:job-definition/*",
                "arn:aws:batch:us-east-2:888888888888:job-queue/queue1"
            ]
        }
    ]
}
```

当所有条件都匹配字符串时拒绝操作

当batch:Image（容器映像 ID）条件密钥均为 “” 且（容器日志驱动程序）条件键均*string1*为 “” 时，以下策略将batch:LogDriver拒绝访问 RegisterJobDefinitionAPI 操作*string2*。” 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。

有关条件密钥的列表 AWS Batch，请参阅《服务授权参考》 AWS Batch中的条件密钥。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。

JSON

```
{
  "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"
        }
      }
    }
  ]
}
```

资源：当所有条件键都匹配字符串时拒绝操作

当batch:Image（容器映像 ID）条件密钥为 “” 或（容器日志驱动程序）条件键为 *string1* “” 时，以下策略拒绝访问 RegisterJobDefinitionAPI 操作*string2*。batch:LogDriver 当作业跨越多个容器（例如多节点 parallel 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。

有关条件密钥的列表 AWS Batch，请参阅《服务授权参考》 AWS Batch中的条件密钥。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。）

JSON

```
{
  "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"
          ]
        }
      }
    }
  ]
}
```

资源：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。

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
      ],
      "Resource": [
        "arn:aws:batch:us-east-2:555555555555:job-definition/JobDefA",
        "arn:aws:batch:us-east-2:555555555555:job-queue/jobqueue1"
      ],
      "Condition": {
        "StringEquals": {
          "batch:ShareIdentifier": [
            "lowCpu"
          ]
        }
      }
    }
  ]
}
```

使用管理 SageMaker AI 资源 AWS Batch

此策略 AWS Batch 允许管理 SageMaker AI 资源。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
```

```
            ],
            "Resource": "arn:aws:iam::*:role/
*AWSServiceRoleForAWSBatchWithSagemaker",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "sagemaker-
queuing.batch.amazonaws.com"
                }
             }
         },
         {
             "Effect": "Allow",
             "Action": "iam:PassRole",
             "Resource": "*",
             "Condition": {
                 "StringEquals": {
                     "iam:PassedToService": [
                         "sagemaker.amazonaws.com"
                      ]
                  }
              }
          }
    ]
}
```

资源：通过作业定义和作业队列上的资源标签限制作业提交

只有在作业队列都有标签`Environment=dev`且作业定义都有标签时，才使用以下策略提交作业`Project=calc`。此策略演示了如何在提交作业期间使用资源标签来控制对 AWS Batch 资源的访问权限。

> **Important**
>
> 使用评估作业定义资源标签的策略提交作业时，必须使用作业定义修订格式（`job-definition:revision`）提交作业。如果您在未指定修订版本的情况下提交作业，则不会评估作业定义标签，这可能会绕过您预期的访问控制。资源 ARN 中的`*:*`模式强制要求提交内容必须包含修订版，从而确保标签策略始终得到有效应用。

此策略使用两个单独的语句，因为它对不同的资源类型应用不同的标签条件。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:*:*:job-queue/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "dev"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "batch:SubmitJob",
      "Resource": "arn:aws:batch:*:*:job-definition/*:*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "calc"
        }
      }
    }
  ]
}
```

资源： AWS Batch 托管策略

AWS Batch 提供了可以附加到用户的托管策略。此政策提供使用 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

此策略允许管理员具有完全访问权限 AWS Batch。

要查看策略的 JSON，请参阅AWS 托管策略参考指南AWSBatchFullAccess中的。

AWS Batch IAM 执行角色

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。

IAM 执行角色是必需的，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。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    }
  ]
}
```

API 操作支持的资源级权限 AWS Batch

“资源级权限” 一词是指能够指定允许用户对其执行操作的资源。 AWS Batch 部分支持资源级权限。对于某些 AWS Batch 操作，您可以根据必须满足的条件来控制何时允许用户使用这些操作。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。

有关 B AWS atch 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》AWS Batch中的操作、资源和条件密钥。 ARNs

教程：创建 IAM 执行角色

如果您的账户尚未具有 IAM 执行角色，请使用以下步骤创建角色。

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色。
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，请选择 Elastic Container Service。然后再次选择 Elastic Container Service 任务。
6. 选择下一步。
7. 要查看权限策略，请搜索 A mazon ECSTask ExecutionRolePolicy。
8. 选中 Amazon ECSTask ExecutionRolePolicy 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。

教程：检查 IAM 执行角色

使用以下过程检查并确定您的账户是否已拥有 IAM 执行角色并且已附加托管 IAM 策略（如果需要）。

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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，请参阅教程：创建 IAM 执行角色。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 Amazon ECSTask ExecutionRolePolicy 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ECSTaskExecutionRolePolicy。

c. 选中 Amazon ECSTask ExecutionRolePolicy 政策左侧的复选框并选择附加政策。

5. 选择信任关系。

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。

JSON

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

将服务关联角色用于 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。

AWS Batch 使用两个不同的服务相关角色：

- AWSServiceRoleForBatch-用于包括计算环境在内的 AWS Batch 操作。
- AWSServiceRoleForAWSBatchWithSagemaker-用于 SageMaker AI 工作负载管理和队列。

主题

- 将角色用于 AWS Batch

- 在 SageMaker AI 中 AWS Batch 使用角色

将角色用于 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他 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，您才能删除服务关联角色。这样可以保护您的 AWS Batch 资源，因为您不会无意中删除访问资源的权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并在服务相关角色列中查找标有 “是” 的服务。选择是和链接，查看该服务的服务关联角色文档。

的服务相关角色权限 AWS Batch

AWS Batch 使用名为的服务相关角色 AWSServiceRoleForBatch— AWS Batch 允许代表您创建和管理 AWS 资源。

AWSServiceRoleForBatch 服务相关角色信任以下服务来代入该角色：

- `batch.amazonaws.com`

名为的角色权限策略BatchServiceRolePolicy AWS Batch 允许对指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。

您必须配置使用户、组或角色能够创建、编辑或删除服务相关角色的权限。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 创建服务关联角色 AWS Batch

您无需手动创建服务关联角色。当您在 AWS 管理控制台、或 AWS API 中创建计算环境时， AWS Batch 会为您创建服务相关角色。 AWS CLI

> **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，此服务关联角色可以出现在您的账户中。如果您在 2021 年 3 月 10 日 AWS Batch 服务开始支持服务相关角色之前使用该服务，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” 中出现了一个新角色 AWS 账户。

如果您删除该服务关联角色，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。创建计算环境时， AWS Batch 会再次为您创建服务相关角色。

为 AWS Batch编辑服务关联角色

AWS Batch 不允许您编辑 AWSServiceRoleForBatch 服务相关角色。创建服务关联角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务关联角色。

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 AWS Batch的服务关联角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能手动删除它。

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. 使用 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 管理控制台 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. 要提交服务相关角色的删除请求，请调用 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. 要检查删除的状态，请调用 GetServiceLinkedRoleDeletionStatus。在请求中，指定 `DeletionTaskId`。

   删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。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 服务相关角色的受支持区域

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。

在 SageMaker AI 中 AWS Batch 使用角色

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，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务关联角色。这样可以保护您的 AWS Batch 资源，因为您不会无意中删除访问资源的权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的AWS 服务，并在服务相关角色列中查找标有 “是” 的服务。选择是和链接，查看该服务的服务关联角色文档。

的服务相关角色权限 AWS Batch

AWS Batch 使用名为 AWSServiceRoleForAWSBatchWithSagemaker— AWS Batch 允许代表您排队和管理 SageMaker 培训作业的服务相关角色。

AWSServiceRoleForAWSBatchWithSagemaker 服务相关角色信任以下服务来代入该角色：

- `sagemaker-queuing.batch.amazonaws.com`

角色权限策略 AWS Batch 允许对指定资源完成以下操作：

- `sagemaker`— AWS Batch 允许管理 SageMaker 训练作业、转换作业和其他 SageMaker AI 资源。
- `iam:PassRole`— AWS Batch 允许将客户定义的执行角色传递给 SageMaker AI 以执行作业。资源限制允许将角色传递给 SageMaker AI 服务。

您必须配置使用户、组或角色能够创建、编辑或删除服务相关角色的权限。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 创建服务关联角色 AWS Batch

您无需手动创建服务关联角色。当您使用`CreateServiceEnvironment` AWS 管理控制台、或 AWS API 创建服务环境时， AWS Batch 会为您创建服务相关角色。 AWS CLI

如果您删除该服务关联角色，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。使用创建服务环境时`CreateServiceEnvironment`， AWS Batch 会再次为您创建服务相关角色。

要查看策略的 JSON，请参阅AWS 托管策略参考指南AWSBatchServiceRolePolicyForSageMaker中的。

为 AWS Batch编辑服务关联角色

AWS Batch 不允许您编辑 AWSServiceRoleForAWSBatchWithSagemaker 服务相关角色。创建服务关联角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

删除 AWS Batch的服务关联角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能手动删除它。

清除服务相关角色

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服务环境。

检查服务相关角色是否有活动会话

1. 使用 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 AWSServiceRoleForAWSBatchWithSagemaker 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 AWSServiceRoleForAWSBatchWithSagemaker 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 AWSServiceRoleForAWSBatchWithSagemaker 服务相关角色使用的 AWS Batch 资源

必须将所有作业队列与所有服务环境断开关联，然后必须先删除所有 AWS 区域中使用该 AWSServiceRoleForAWSBatchWithSagemaker 角色的所有服务环境，然后才能删除该 AWSServiceRoleForAWSBatchWithSagemaker 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，选择环境，然后选择服务环境。
4. 选择所有服务环境。
5. 选择 Disable (禁用)。等待状态变为已禁用。
6. 选择该服务环境。
7. 选择删除。选择删除服务环境，以确认您确实要删除该服务环境。
8. 在所有区域中对使用该服务相关角色的所有服务环境重复第 1–7 步。

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 管理控制台 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 AWSService RoleFor AWSBatchWithSagemaker，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
 AWSServiceRoleForAWSBatchWithSagemaker
```

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. 要提交服务相关角色的删除请求，请调用 DeleteServiceLinkedRole。在请求中，指定 AWSServiceRoleForAWSBatchWithSagemaker 角色名称。

   如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `DeletionTaskId` 以检查删除任务的状态。

2. 要检查删除的状态，请调用 GetServiceLinkedRoleDeletionStatus。在请求中，指定 `DeletionTaskId`。

   删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。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 服务相关角色的受支持区域

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。

Amazon ECS 实例角色

AWS Batch 计算环境中填充了 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 Amazon ECS 实例角色。

主题

检查您账户的 Amazon ECS 实例角色

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。

教程：在 IAM 控制台中检查 **`ecsInstanceRole`**

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择角色。
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. 选择创建角色。
    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. 选择下一步。
    e. 要查看权限策略，请搜索 A mazon EC2 ContainerServicefor EC2 角色。
    f. 选中 Amazon EC2 ContainerServicefor EC2 角色旁边的复选框，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**
>
> 如果您使用 AWS 管理控制台 为 Amazon EC2 创建角色，则控制台会创建与该角色同名的实例配置文件。

或者，您可以使用创建 `ecsInstanceRole` IAM 角色。 AWS CLI 以下示例创建了一个带有信任策略和 AWS 托管策略的 IAM 角色。

教程：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。

   JSON

```
{
  "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
```

3. 使用create-instance-profile命令创建名为的实例配置文件`ecsInstanceRole`。

> **Note**
>
> 您需要在和 AWS API 中创建角色和实例配置文件作为单独的 AWS CLI 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-profil e 命令将ecsInstanceRole角色添加到ecsInstanceRole实例配置文件中。

```
aws iam add-role-to-instance-profile \
    --role-name ecsInstanceRole --instance-profile-name ecsInstanceRole
```

5. 使用attach-role-policy命令将AmazonEC2ContainerServiceforEC2Role AWS 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。您还必须拥有 Amazon EC2 AWSServiceRoleForEC2竞价AWSServiceRoleForEC2SpotFleet型和竞价型队列的竞价和服务相关角色。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角色向 AWS 服务委派权限。

主题

- 在 AWS 管理控制台中创建 Amazon EC2 竞价型实例集角色
- 使用创建 Amazon EC2 竞价型队列角色 AWS CLI

在 AWS 管理控制台中创建 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. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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. 对于其他用例 AWS 服务，请选择 EC2，然后选择 EC 2-Spot 队列标记。
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。
>
> - Amazon EC2 SpotFleetRole：这是 Spot 队列角色的原始托管策略。但是，我们不再建议您将其与一起使用 AWS Batch。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。

- 亚马逊 EC2 SpotFleetTaggingRole：此角色提供标记 Amazon EC2 竞价型实例的所有必要权限。使用此角色允许在您的 AWS Batch 计算环境上标记 Spot 实例。

使用创建 Amazon EC2 竞价型队列角色 AWS CLI

为您的 Spot 队列计算环境创建 EC2 SpotFleetTaggingRole A mazon 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. 要将亚马逊EC2SpotFleetTaggingRole托管 IAM 策略附加到您的亚马逊EC2SpotFleetTaggingRole角色，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 \
    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 提供了描述 AWS 资源变化的近乎实时的系统事件流。 AWS Batch 可以将工作作为 EventBridge 目标。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。在提交带有 EventBridge 规则和目标的 AWS Batch 作业之前， EventBridge 必须具有代表您运行 AWS Batch 作业的权限。

> **Note**
>
> 在 EventBridge 控制台中创建将 AWS Batch 队列指定为目标的规则时，可以创建此角色。有关示例演练的信息，请参阅AWS Batch 作业作为 EventBridge 的目标。您可以使用 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）。

您的 EventBridge IAM 角色的信任关系必须为`events.amazonaws.com`服务委托人提供担任该角色的能力。

JSON

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

确保附加到您的 EventBridge IAM 角色的策略允许`batch:SubmitJob`访问您的资源。在以下示例中， AWS Batch 给出了提供这些权限的`AWSBatchServiceEventTargetRole`托管策略。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
```

```
        ],
        "Resource": "*"
      }
    ]
}
```

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（亚马逊 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。

Option	值
要创建的资源	仅限 VPC
Name	可以选择为您的 VPC 提供名称。
IPv4 CIDR 块	IPv4 CIDR 手动输入 CIDR 数据块大小必须在 /16 和 /28 之间。
IPv6 CIDR 块	没有 IPv6 CIDR 块
Tenancy	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？。

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 AWS Batch 托管计算环境，将计算资源启动到您的新 VPC 中。有关更多信息，请参阅 创建计算环境。如果您在 AWS Batch 控制台中使用计算环境创建向导，则可以指定刚刚创建的 VPC 以及要启动实例的公有或私有子网。
- 创建映射到您的新计算环境的 AWS Batch 任务队列。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 教程：提交作业。

使用接口终端节点进行访问 AWS Batch

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 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
- com.amazonaws。 *region*.batch-fips（有关符合 FIPS 标准的终端节点，请参阅终端节点和配额）AWS Batch

例如：

```
com.amazonaws.us-east-2.batch
```

```
com.amazonaws.us-east-2.batch-fips
```

在aws-cn分区中，格式不同：

```
cn.com.amazonaws.region.batch
```

例如：

```
cn.com.amazonaws.cn-northwest-1.batch
```

AWS Batch 接口终端节点的私有 DNS 名称

如果您为接口终端节点启用私有 DNS，则可以使用特定的 DNS 名称进行连接 AWS Batch，我们提供以下选项：

- 批量。 *region*.amazonaws.co
- 批量。 *region*.api.aws

对于符合 FIPS 标准的端点：

- 批量翻转。 *region*.api.aws
- fips.batch。 *region*不支持.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。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

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。

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）终端节点或特定终端节点的访问 VPCs。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

防止跨服务混淆代理

混淆代理问题是一个安全问题，即没有执行操作权限的实体可能会迫使更具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：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。

JSON

```
{
  "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：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。

JSON

```
{
```

```
  "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/*"
          ]
        }
      }
    }
  ]
}
```

使用记录 AWS Batch API 调用 AWS CloudTrail

AWS Batch 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在中执行的操作的记录 AWS Batch。 CloudTrail 将所有 API 调用捕获 AWS Batch 为事件。捕获的调用包括来自 AWS Batch 控制台的调用和对 AWS Batch API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括的事件 AWS Batch。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 AWS Batch、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅AWS CloudTrail 用户指南。

主题

- AWS Batch 信息在 CloudTrail
- 参考：了解 AWS Batch 日志文件条目

AWS Batch 信息在 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当活动发生在中时 AWS Batch，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您 AWS 账户中的事件，包括的事件 AWS Batch，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，当您在控制台中创建跟踪时，该跟踪将应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

所有 AWS Batch 操作都由记录 CloudTrail 并记录在 https://docs.aws.amazon.com/batch/latestAPIReference//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用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

参考：了解 AWS Batch 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、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，因此它们不会以任何特定顺序显示。

以下示例显示了演示该`CreateComputeEnvironment`操作的 CloudTrail 日志条目。

```
{
  "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"
}
```

排查 AWS Batch IAM

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 管理控制台 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，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

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访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

AWS Step Functions

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

教程：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。

教程：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。

教程：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。

Amazon EventBridge 的 AWS Batch 事件流

可以使用 Amazon EventBridge AWS Batch事件流近乎实时地接收通知，了解在作业队列中的作业的当前状态。

可以使用 EventBridge 来进一步获得有关AWS Batch服务的见解。更具体地说，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。借助 AWS Batch和 EventBridge，无需计划和监控用于持续轮询AWS Batch以了解作业状态更改的代码。相反，可以使用各种 Amazon EventBridge 目标异步处理AWS Batch作业状态的变化。其中包括 AWS Lambda、Amazon Simple Queue Service、Amazon Simple Notification Service 或 Amazon Kinesis Data Streams。

确保传送AWS Batch事件流中的事件至少一次。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作业可作为 EventBridge 的目标提供。可以通过使用简单的规则来匹配事件并根据事件提交 AWS Batch 作业。有关更多信息，请参阅 Amazon EventBridge 用户指南中的什么是 EventBridge？ 还可以使用 EventBridge 计划自动化操作，这些操作可在特定时间使用cron或 rate 表达式自动触发。有关更多信息，请参阅 Amazon EventBridge 用户指南中的创建按计划运行的 Amazon EventBridge 规则 有关示例演练的信息，请参阅AWS Batch 作业作为 EventBridge 的目标。有关 EventBridge 调度器的更多信息，请参阅《Amazon EventBridge 用户指南》中的设置 Amazon EventBridge 调度器。

主题

- AWS Batch 事件
- 教程：将 AWS 用户通知用于 AWS Batch
- AWS Batch 作业作为 EventBridge 的目标
- 教程：使用 EventBridge 侦听 AWS Batch 作业事件
- 教程：针对作业失败事件发送 Amazon Simple Notification Service 警报

AWS Batch 事件

AWS Batch将作业状态更改事件发送到 EventBridge。AWS Batch会跟踪作业的状态。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、来源和详细信息。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，请参阅任务状态。

> **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。`detail` 部分类似于 AWS BatchAPI Reference 中的 DescribeJobs API 操作返回的 JobDetail 对象。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

```
{
    "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": []
        },
        "propagateTags": false,
        "platformCapabilities": []
    }
}
```

作业队列阻塞事件

每当 AWS Batch 检测到作业处于 RUNNABLE 状态从而阻止队列时，都会在 Amazon CloudWatch Events 中创建一个事件。有关支持的队列阻止原因的更多信息，请参阅 作业在RUNNABLE状态卡住。同样的原因也提供在 DescribeJobs API 操作的 statusReason 字段中。

Example作业队列阻塞事件

作业队列阻塞事件的传输格式如下。`detail` 部分类似于 AWS BatchAPI Reference 中的 DescribeJobs API 操作返回的 JobDetail 对象。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

```
{
    "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": []
        },
        "propagateTags": false,
        "platformCapabilities": []
    }
}
```

服务作业状态更改事件

只要现有服务作业状态发生更改，就会创建一个事件。有关服务作业状态的更多信息，请参阅将 AWS Batch 服务作业状态映射到 SageMaker AI 状态。

> **Note**
>
> 对于初始作业提交不会创建事件。

Example服务作业状态更改事件

服务作业状态更改事件的传输格式如下。`detail` 部分与《AWS Batch API 参考》中 DescribeServiceJob API 操作返回的响应相同。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

> **Note**
>
> `detail` 事件不包含 `tags` 和 `serviceRequestPayload` 字段。

```
{
  "version": "0",
  "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
  "detail-type": "Batch Service 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:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8"
  ],
  "detail": {
    "attempts": [
      {
        "serviceResourceId": {
          "name": "TrainingJobArn",
          "value": "arn:aws:sagemaker:us-east-1:123456789012:training-job/AWSBatchmy-
training-job88b610a69aa8380ca5b0a7aba3f81cb8"
        },
        "startedAt": 1641944300058,
        "stoppedAt": 1641944400058,
        "statusReason": "Received status from SageMaker: Training job completed"
      }
    ],
    "createdAt": 1641944200058,
    "jobArn": "arn:aws:batch:us-east-1:123456789012:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
    "jobId": "0bb17543-ece6-4480-b1a7-a556d344746b",
    "jobName": "event-test",
    "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/HappyPathJobQueue",
    "latestAttempt": {
      "serviceResourceId": {
        "name": "TrainingJobArn",
        "value": "arn:aws:sagemaker:us-east-1:123456789012:training-job/AWSBatchmy-
training-job88b610a69aa8380ca5b0a7aba3f81cb8"
      }
    },
    "serviceJobType": "SAGEMAKER_TRAINING",
    "startedAt": 1641944300058,
    "status": "SUCCEEDED",
    "statusReason": "Received status from SageMaker: Training job completed",
    "stoppedAt": 1641944400058,
```

```
    "timeoutConfig": {
      "attemptDurationSeconds": 60
    }
  }
}
```

服务作业队列阻塞事件

每当 AWS Batch 检测到处于阻塞状态的队列时，都会在 Amazon CloudWatch Events 中创建一个事件。队列阻塞的原因也会在 DescribeServiceJob API 操作的 statusReason 字段中提供。

Example服务作业队列阻塞事件

服务作业队列阻塞事件的传输格式如下。detail 部分与《AWS Batch API 参考》中 DescribeServiceJob API 操作返回的响应相同。有关 EventBridge 参数更多信息，请参阅 Amazon EventBridge 用户指南中事件和事件模式。

> **Note**
>
> detail 事件不包含 tags 和 serviceRequestPayload 字段。

```
{
  "version": "0",
  "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
  "detail-type": "Batch Service 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:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8"
  ],
  "detail": {
    "attempts": [],
    "createdAt": 1641944200058,
    "jobArn": "arn:aws:batch:us-east-1:123456789012:service-job/4c7599ae-0a82-49aa-
ba5a-4727fcce14a8",
    "jobId": "6271dfdf-d8a7-41b1-a4d2-55a2224f5375",
    "jobName": "event-test",
    "jobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/HappyPathJobQueue",
```

```
    "serviceJobType": "SAGEMAKER_TRAINING",
    "status": "RUNNABLE",
    "statusReason": "blocked-reason",
    "timeoutConfig": {
      "attemptDurationSeconds": 60
    }
  }
}
```

教程：将 AWS 用户通知用于 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。您可以通过多个渠道接收事件通知，包括电子邮件、聊天应用程序中的 Amazon Q 开发者版聊天通知或 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AWS Batch中配置用户通知：

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，请参阅用户通知AWS入门。

AWS Batch 作业作为 EventBridge 的目标

Amazon EventBridge 提供近乎实时的系统事件流，这些系统事件可以描述亚马逊云科技资源中的更改。通常，在 Amazon Elastic Container Service，Amazon Elastic Kubernetes Service 和 AWS Fargate 作业上的 AWS Batch 作为 EventBridge 的目标提供。可以通过使用简单的规则来匹配事件并根据事件提交 AWS Batch 作业。有关更多信息，请参阅 Amazon EventBridge 用户指南中的什么是 EventBridge？

您还可以使用 EventBridge 来计划使用 cron 或 rate 表达式在某些时间触发的自动化操作。有关更多信息，请参阅 Amazon EventBridge 用户指南中的创建按计划运行的 Amazon 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Amazon EventBridge 用户指南中的创建对事件作出反应的 Amazon EventBridge 规则。

作为 EventBridge 目标的 AWS Batch 作业的常见用例包括以下：

- 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 作业为了响应 CloudTrail 中记录的 API 操作而运行。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**
>
> 在本场景中，所有相关 AWS 资源必须位于同一区域中。这包括诸如 Amazon S3 存储桶、EventBridge 规则和 CloudTrail 日志之类的资源。

在您可以使用 EventBridge 规则和目标提交 AWS Batch 作业之前，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 作业指定为目标的规则时，您可以创建此角色。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。

主题

- 教程：创建计划的 AWS Batch 作业
- 教程：创建一个具有事件模式的规则
- 教程：使用 EventBridge 输入转换器按计划将事件信息传递给 AWS Batch 目标

教程：创建计划的 AWS Batch 作业

以下过程介绍了如何创建计划的 AWS Batch 作业和所需的 EventBridge IAM 角色。

若要使用 EventBridge 创建计划 AWS Batch 任务

> **Note**
>
> 此过程适用于 Amazon ECS、Amazon EKS 和 AWS Fargate 作业上所有的 AWS Batch。

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 从导航栏中，选择要使用的AWS 区域。

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。
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 job queue（批处理作业队列）目标类型，EventBridge 需要权限才能将事件发送到目标。EventBridge 可以创建运行事件所需的 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 Amazon EventBridge 用户指南中的创建按计划运行的 Amazon EventBridge 规则

教程：创建一个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。

创建一条规则，在事件与定义的模式匹配时将事件发送到目标

> **Note**
> 此过程适用于 Amazon ECS、Amazon EKS 和 AWS Fargate 作业上所有的 AWS Batch。

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。

2. 从导航栏中，选择要使用的AWS 区域。
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。
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. 对于事件源，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 job queue（批处理作业队列）目标类型，EventBridge 需要权限才能将事件发送到目标。EventBridge 可以创建运行事件所需的 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《Amazon EventBridge 用户指南》中的 Amazon 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。

    要了解如何创建规则的更多信息，请参阅Amazon EventBridge 用户指南中的创建对事件作出反应的 Amazon EventBridge 规则。

教程：使用 EventBridge 输入转换器按计划将事件信息传递给 AWS Batch 目标

您可以使用 EventBridge 输入转换器在任务提交时将事件信息传递至 AWS Batch。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 输入转换器可以根据事件数据提供参数值。

然后，您只需创建一个 AWS Batch 事件目标，该目标将解析来自触发它的事件的信息，并将它转换为 `parameters` 对象。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）都必须位于同一个区域。

创建使用输入转换器的 AWS Batch 目标

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 从导航栏中，选择要使用的AWS 区域。
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 AWS 区域中和同一事件总线上的另一条规则的名称相同。

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。
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 job queue（批处理作业队列）目标类型，EventBridge 需要权限才能将事件发送到目标。EventBridge 可以创建运行事件所需的 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《Amazon EventBridge 用户指南》中的 Amazon 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：使用 EventBridge 侦听 AWS Batch 作业事件

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AWS Batch 教程入门中的步骤创建一个。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。

主题

- 教程：创建 Lambda 函数
- 教程：注册事件规则
- 教程：测试配置

教程：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. 通过 https://console.aws.amazon.com/lambda/ 打开 AWS Lambda 控制台。
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. 选择部署。

教程：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 管理控制台 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。

```
{
  "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. 选择下一步。
12. 对于 Target types（目标类型），选择 AWS service（AWS 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。请执行以下操作之一：

- 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. 选择下一步。
18. 查看规则详细信息并选择创建规则。

教程：测试配置

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 教程：提交作业。
3. 通过 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
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 Amazon EventBridge 事件规则，它只捕获作业已转为 `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AWS Batch 教程入门中的步骤创建一个。

主题

- 教程：创建并订阅 Amazon SNS 主题
- 教程：注册事件规则
- 教程：测试您的规则
- 替代规则：Batch 作业队列被阻止

教程：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。

教程：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. 打开位于 https://console.aws.amazon.com/events/ 的 Amazon EventBridge 控制台。
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"
    ]
  }
}
```

   此代码定义一个与作业状态为 `FAILED` 的任何事件匹配的 EventBridge 事件规则。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. 选择下一步。
12. 对于目标类型，选择AWS 服务。
13. 对于 选择目标，选择 SNS 主题；对于 主题，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 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。请执行以下操作之一：

- 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. 选择下一步。
18. 查看规则详细信息并选择创建规则。

教程：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，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 教程：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 作业队列被阻止

要创建监控批处理作业队列被阻止的事件规则，请重复这些教程，并进行以下更改：

1. 在 教程：创建并订阅 Amazon SNS 主题 中，使用 *BlockedJobQueue* 作为主题名称。
2. 在 教程：注册事件规则 中，在 JSON 编辑器中使用以下模式：

```
{
   "detail-type": [
     "Batch Job Queue Blocked"
   ],
   "source": [
     "aws.batch"
   ]
}
```

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 的实例类型的列表，请参阅《Amazon EC2 用户指南》中的支持的实例类型。

  > **Tip**
  >
  > 要查看AWS 区域中支持 EFA 的实例类型列表，请执行以下命令。然后，将返回的列表与AWS Batch控制台中的可用实例类型列表进行交叉引用。
  >
  > ```
  > $ 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`

  对于多节点并行作业和单节点容器作业，LinuxParameters 成员的位置将不同。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 更多信息，请参阅《Amazon EC2 用户指南》中的 Elastic Fabric Adapter。

监视器 AWS Batch

监控是维护 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS Batch

我们强烈建议您从 AWS 解决方案的各个部分收集监控数据，以便在出现多点故障时更轻松地进行调试。首先创建一个监控计划来回答以下问题。如果您不确定如何回答这些问题，您仍然可以使用 Amazon CloudWatch Logs 来建立绩效基准。

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

下一步是通过测量不同时间和不同负载条件下的性能，来建立环境中正常 AWS Batch 性能的基准。监控时 AWS Batch，请保留历史监控数据，以便可以将其与当前性能数据进行比较。这将帮助您确定一般的性能模式和性能异常，并设计解决问题的方法。

本部分中的主题可帮助您启动对 AWS Batch的日志记录和监控操作。

主题

- 将 CloudWatch 日志与配合使用 AWS Batch
- AWS Batch CloudWatch Container Insights
- 使用 CloudWatch Logs 监控 Amazon EKS 作业的 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 作业，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 “教程：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。您可以添加参数来更改两者 EC2 和 Fargate 资源的默认行为。有关更多信息，请参阅 the section called “使用 awslogs 日志驱动程序”。

主题

教程：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 日志的 IAM 策略 APIs。 CloudWatch 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 日志。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. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

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：

   JSON

```
{
    "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. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
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 CloudWatchLogs 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AWS Batch 教程入门。

> **Note**
>
> AWS Fargate 资源不支持启动模板。

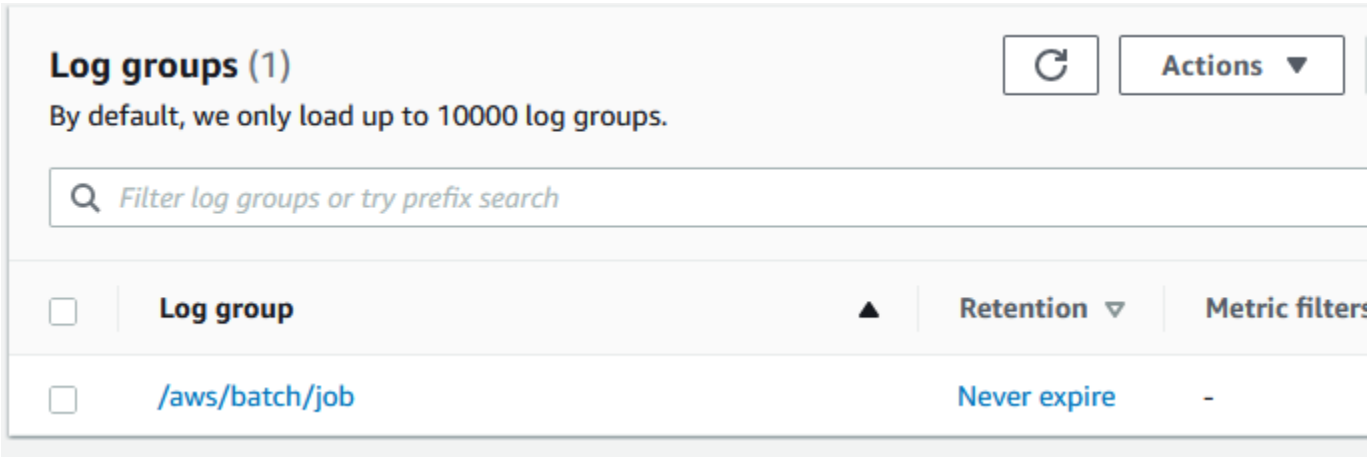
教程：查看 CloudWatch 日志

您可以在中查看和搜索 CloudWatch 日志日志 AWS 管理控制台。

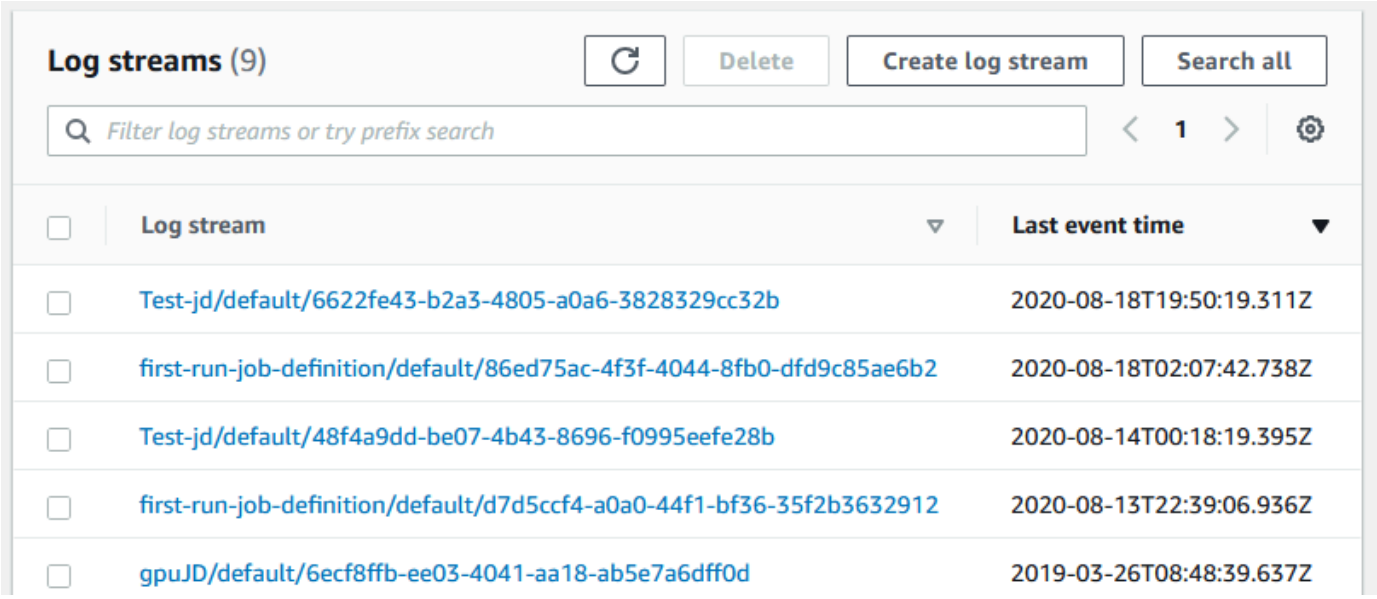
> **Note**
>
> 数据可能需要几分钟才能显示在 “ CloudWatch 日志” 中。

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。

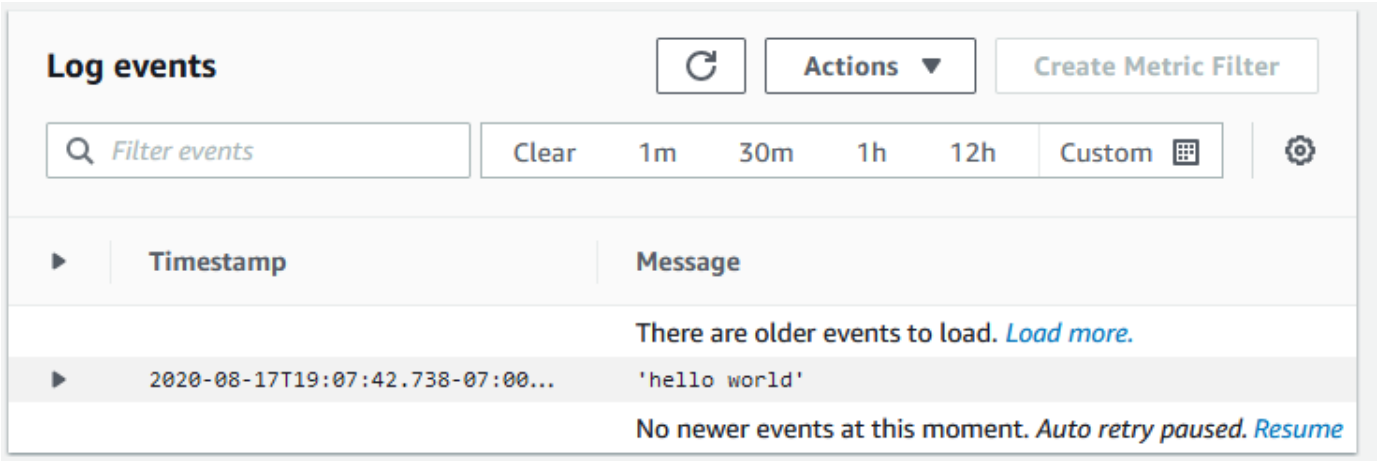


3. 选择要查看的日志组。



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。



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。

主题

开启 Container Insights

完成以下步骤以为 AWS Batch 计算环境开启 Container Insights。

1. 打开AWS Batch控制台。
2. 选择环境。
3. 选择所需的计算环境。
4. 在 Container Insights 选项卡中，为计算环境开启 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 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，请参阅使用 Amazon CloudWatch Observability EKS 附加组件或 Helm 图表安装 CloudWatch 代理。

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。

安装附加组件

有关安装 AWS for Fluent Bit 并创建 CloudWatch 组的说明，请参阅使用 Amazon CloudWatch Observability EKS 附加组件或 Helm 图表安装 CloudWatch 代理。

安装此附加组件时，您必须提供以下额外的配置数据：

- 如果使用 AWS 管理控制台安装此附加组件，则需要在配置值中提供以下容差：

```
{
  "tolerations": [
    {
      "key": "batch.amazonaws.com/batch-node",
      "operator": "Exists"
    }
  ]
}
```

- 如果使用 AWS CLI 安装此附加组件，请添加以下参数：

```
--configuration-values '{"tolerations":[{"key":"batch.amazonaws.com/batch-
node","operator":"Exists"}]}'
```

> **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。

标记 AWS Batch 资源

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。

主题

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 管理控制台、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。

标记资源

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。

AWS Batch 资源的标签支持

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 作业	支持	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。

教程：使用控制台管理标签

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。

在单个资源上添加和删除标签

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。

使用 CLI 或 API 管理标签

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。

AWS Batch 资源的标签支持

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
创建消耗性资源	CreateConsumableResource	create-consumable-resource	Create-BATConsumableResource

AWS Batch 的最佳做法

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。

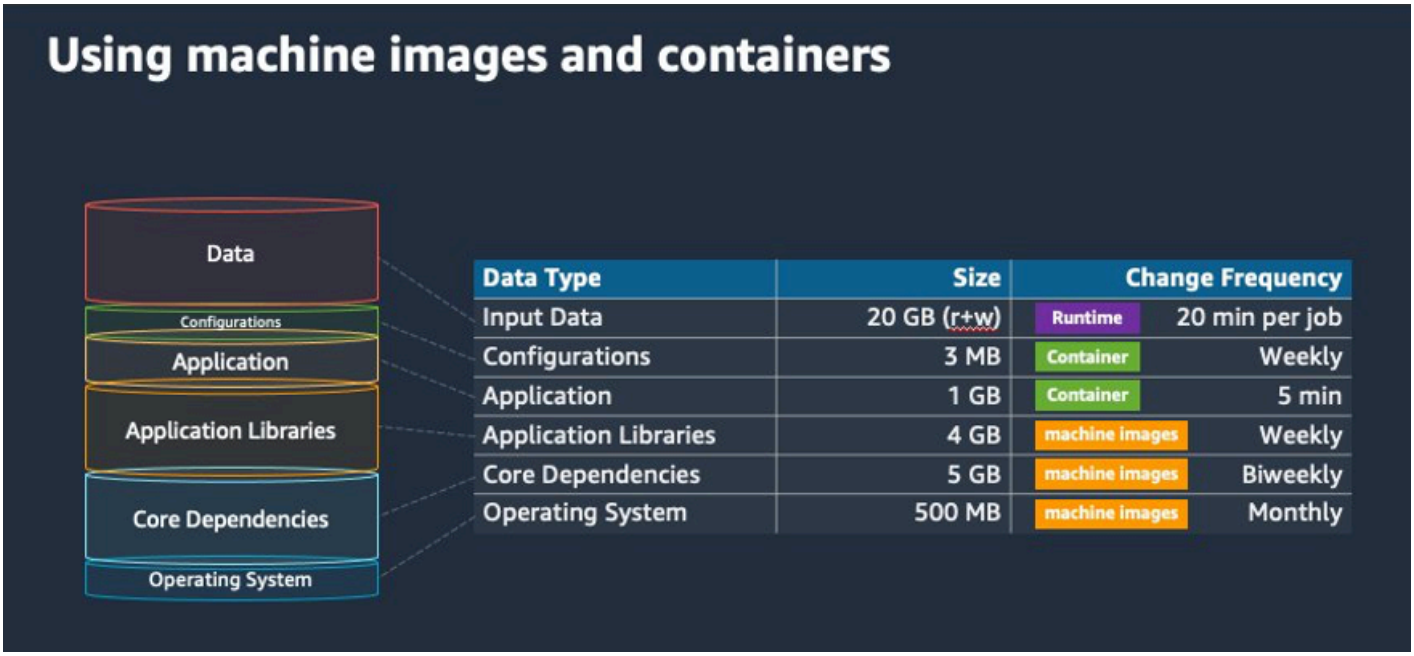
- 查看您的 Amazon EC2 限额 – 在 AWS 管理控制台 的“服务限额”面板中查看您的 Amazon EC2 限额（也称为限制）。如有必要，可以申请增加您的 Amazon EC2 实例峰值数量的限额。请记住，Amazon EC2 竞价型和 Amazon 按需型实例有单独的限额。有关更多信息，请参阅服务限额入门。
- 验证每个区域的 Amazon Elastic Block Store 限额 – 每个实例对操作系统使用 GP2 或 GP3 卷。默认情况下，每个 AWS 区域 限额为 300TiB。但是，每个实例都使用计数作为此限额的一部分。因此，在验证每个区域的 Amazon Elastic 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 4GB，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）。您也可以使用绑定装载，为容器授予访问权限。有关更多信息，请参阅绑定装载。
- 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 2GB，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 120GiB 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 竞价型最佳实践用于 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 竞价型的最佳实践。

要优化您的工作流程以节省成本，请考虑以下 AWS Batch 的 Amazon EC2 竞价型最佳实践：

- 选择 **SPOT_CAPACITY_OPTIMIZED** 分配策略 – AWS Batch 从最深的 Amazon EC2 竞价型容量池中选择 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 AWS Batch 的实例类型分配策略。
- 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 竞价型中断的信息，请参阅竞价型中断控制面板。

  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"
    }]
}
```

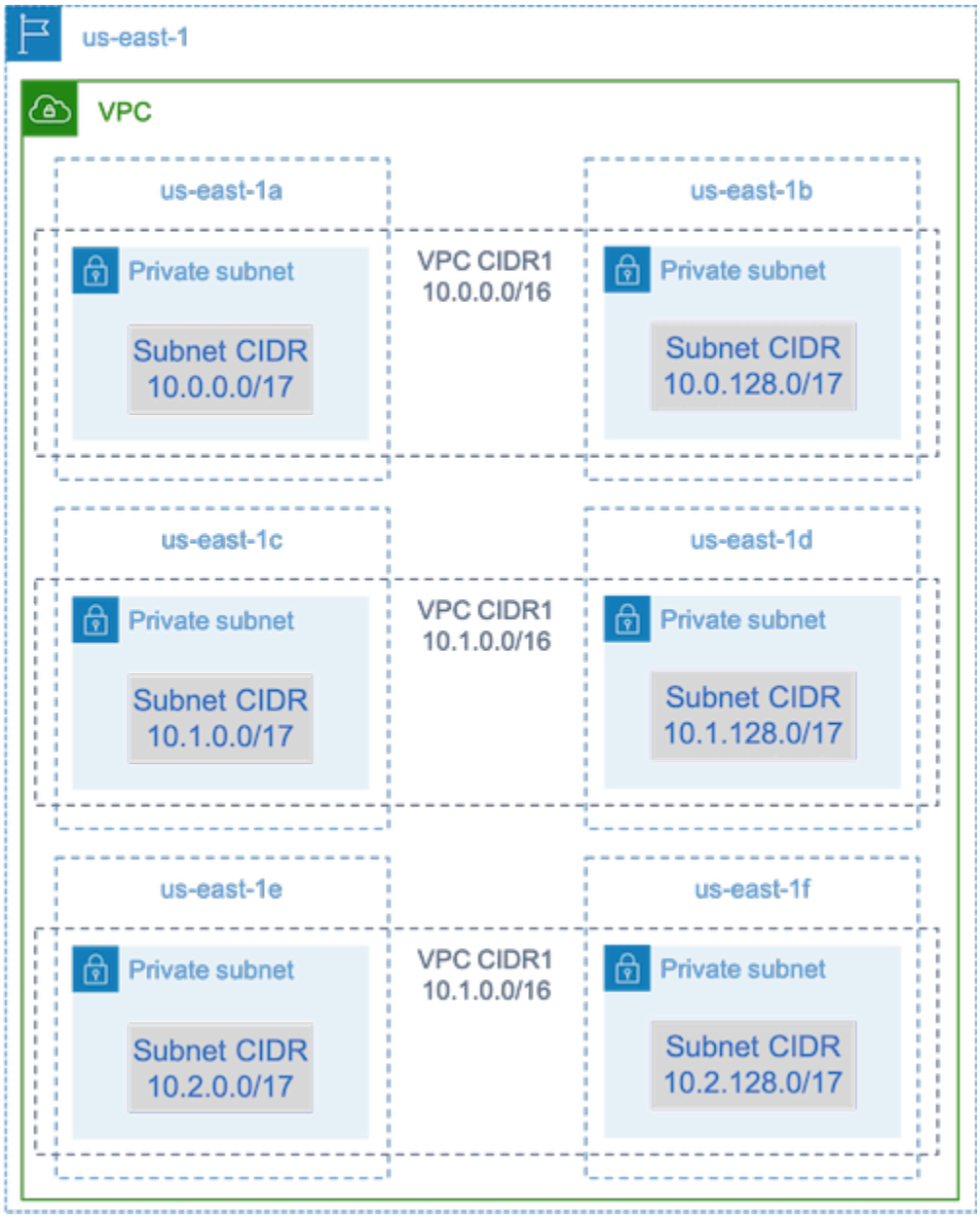
- 使用竞价型中断控制面板 – 您可以使用竞价型中断控制面板来跟踪竞价型中断情况。应用程序提供有关已回收的 Amazon EC2 竞价型实例以及竞价型实例所在的可用区的指标。有关更多信息，请参阅竞价型中断控制面板

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 竞价型 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。如果您实例尚未创建，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 竞价型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 支持 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。

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、亚马逊 ECS 和亚马 EC2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

请查看以下主题，查找审核流程和使用 AWS Batch时可能遇到的常见问题的潜在解决方案。

主题

用于接收自动实例系列更新的最佳实例类型配置

> **Note**
>
> 从 2025 年 11 月 1 日起，`optimal` 的行为将更改为匹配 `default_x86_64`。在更改期间，您的实例系列可能会更新到新一代实例。本次升级无需您执行任何操作。

AWS Batch 支持 Instan ceTypes 中的单个选项`optimal`来满足任务队列的需求。我们引入了两个新的实例类型选项：`default_x86_64` 和 `default_arm64`。如果您未选择任何实例类型，我们将使用

default_x86_64。启用这些新选项后，系统将根据作业队列的要求自动跨系列和代系选择经济实惠的实例类型，让工作负载能够快速运行。

当新的实例类型有足够的容量可用时 AWS 区域，相应的默认池将自动使用新的实例类型进行更新。现有的 optimal 选项将继续受支持，不会弃用，因为底层默认池将支持该选项，以在未来提供更新后的实例。如果您使用的是 optimal 选项，则无需执行任何操作。

但是，请注意，只有ENABLEDVALID计算环境 (CEs) 会自动使用新的实例类型进行更新。如果您有DISABLED或 INVALID CEs，则它们将在重新启用并设置为VALID状态后收到更新。

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。

> **Important**
>
> AWS Batch 代表您并在您的账户中创建和管理多个 AWS 资源，包括亚马逊 EC2 启动模板、Amazon A EC2 uto Scaling 群组、Amazon EC2 Spot 队列和 Amazon ECS 集群。这些托管式资源已特别为了确保最优的 AWS Batch 操作而进行配置。除非 AWS Batch 文档中明确说明，否则手动修改这些由 Batch 托管的资源可能会导致 INVALID 计算环境中的异常行为、不理想的实例扩缩行为、工作负载处理延迟，或产生意外的成本。该服务无法确定性地支持这些手动修改。 AWS Batch 请务必使用支持的 Batch APIs 或 Batch 控制台来管理您的计算环境。

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 或创建的计算环境中更为常见 AWS SDKs。当您在中创建计算环境时 AWS 管理控制台， 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，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。只有在使用 AWS CLI 或时才指定 IAM 角色的名称 AWS SDKs，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，如果使用将服务关联角色用于 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 或时仅指定 IAM 角色的名称 AWS SDKs，则 AWS Batch 假设您的 ARN 不使用`aws-service-role`路径前缀。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。

   > **Tip**
   >
   > 有关如何创建服务相关角色的说明，请参阅将角色用于 AWS Batch。

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置在计算资源上，并导致您的作业队列被阻止。以下介绍了如何了解您的作业是在等待轮候还是被卡住并阻止了队列。

如果 AWS Batch 检测到您的头部有RUNNABLE工作并封锁了队列，您将收到来自 Amazon E 作业队列阻塞事件 vents CloudWatch 的事件，其中包含原因。同样的原因也会更新到 `statusReason` 字段，作为 `ListJobs` 和 `DescribeJobs` API 调用的一部分。

或者，您可以通过 `CreateJobQueue` 和 `UpdateJobQueue` API 操作配置 `jobStateTimeLimitActions` 参数。

> **Note**
>
> 目前，您可以用于 `jobStateLimitActions.action` 的唯一操作是取消作业。

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 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 RUNNABLE 状态等待足够容量可用的任何作业，您可以将 jobStateTimeLimitActions 参数设置为最多等待 4 小时。为此，您可以先将 statusReason 设置为 CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY 并将 maxTimeSeconds 设置为 144000，然后再取消作业并允许下一个作业进入作业队列的开头。

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出现容量不足错误。根据请求， AWS Batch 检测出现容量不足错误的 Amazon EC2 实例。手动取消作业可让后续作业移至队列开头，但如果服务角色问题未得到解决，则很可能下一个作业也将被阻止。最好手动调查并解决此问题。
   - 作业卡住时的 **statusReason** 消息：CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY - Service cannot fulfill the capacity requested for instance type [instanceTypeName]
   - 用于 **jobStateTimeLimitActions** 的 **reason**：CAPACITY:INSUFFICIENT_INSTANCE_CAPACITY
   - 作业取消后的 **statusReason** 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 将服务关联角色用于 AWS Batch 服务相关角色（SLR）或 AWS 托管策略:AWSBatchServiceRole策略 托管式策略，则无需执行任何操作，因为其权限策略已更新。

   b. 如果您使用 SLR 或托管策略，则必须添加 autoscaling:DescribeScalingActivities 和 ec2:DescribeSpotFleetRequestHistory 权限，以便在处于 RUNNABLE 时可以接收已阻止的作业队列事件和更新的作业状态。此外， AWS Batch 需要这些权限才能通过 jobStateTimeLimitActions 参数执行 cancellation 操作，即使它们在作业队列中进行了配置。

   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：所有计算环境都具有小于作业要求的 maxvCpus 参数。手动或通过设置 statusReason 上的 jobStateTimeLimitActions 参数取消作业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 参数以满足已阻止作业的需求。

   - 作业卡住时的 **statusReason** 消息：MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE - CE(s) associated with the job queue cannot meet the CPU requirement of the job.
   - 用于 **jobStateTimeLimitActions** 的 **reason**：MISCONFIGURATION:COMPUTE_ENVIRONMENT_MAX_RESOURCE
   - 作业取消后的 **statusReason** 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作业要求的实例。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。手动或通过设置 statusReason 上的 jobStateTimeLimitActions 参数取消作业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所允许的实例类型，以添加必要的作业资源。

   - 作业卡住时的 **statusReason** 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.
   - 用于 **jobStateTimeLimitActions** 的 **reason**：MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT
   - 作业取消后的 **statusReason** 消息：Canceled by JobStateTimeLimit action due to reason: MISCONFIGURATION:JOB_RESOURCE_REQUIREMENT

4. 原因：所有计算环境都有服务角色问题。要解决此问题，请将您的服务角色权限与 AWS 的托管策略 AWS Batch 进行比较并设法解决任何差距。注意：此错误无法通过 jobStateTimeLimitActions 参数配置可编程操作来解决。

   最佳做法是使用 将服务关联角色用于 AWS Batch 来避免类似的错误。

   手动或通过设置 statusReason 上的 jobStateTimeLimitActions 参数取消作业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，则下一个作业也可能被阻止。最好手动调查并解决此问题。

   - 作业卡住时的 **statusReason** 消息：MISCONFIGURATION:SERVICE_ROLE_PERMISSIONS – Batch service role has a permission issue.

5. 原因：您的计算环境具有不受支持的实例类型配置。当实例类型在您选择的可用区中不可用，或者您的启动模板或启动配置包含与指定实例类型不兼容的设置时，即可能会出现此问题。要解决这个问题，请验证您的指定 AWS 区域 和可用区是否支持您的实例类型，检查您的启动模板设置是否与

您的实例类型兼容，并考虑更新到新一代的实例类型。有关查找支持的实例类型的更多信息，请参阅亚马逊 EC2 用户指南中的查找亚马逊 EC2 实例类型。

- 作业卡住时的 **statusReason** 消息：`MISCONFIGURATION:EC2_INSTANCE_CONFIGURATION_UNSUPPORTED - Your compute environment associated with this job queue has an unsupported instance type configuration.`

6. 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。
   - 作业卡住时的 **statusReason** 消息：`ACTION_REQUIRED - CE(s) associated with the job queue are invalid.`
7. 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。有关故障排除的更多信息，请参阅 re:Post 中的为什么我的 AWS Batch 作业在 AWS上卡在 RUNNABLE 状态。
   - 作业卡住时的 **statusReason** 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，以下是导致此问题的一些常见原因。

计算资源上未配置 **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; 规范和教程：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。

无法通过互联网访问计算资源

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 实例数量取决于您的 EC2 实例配额。 AWS 区域 某些实例类型也有配 per-instance-type额。有关您账户的亚马逊 EC2 实例配额的更多信息，包括如何申请提高限额，请参阅亚马逊 EC2 用户指南中的亚马逊 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。默认情况下，Amazon ECS 容器代理安装在经过优化的亚马逊 ECS 上 AMIs。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

从 2017 年 10 月 25 日起支持对AWS Batch计算资源标记竞价型实例。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 AmazonEC2SpotFleetTaggingRole，然后选择附上策略。
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 AmazonEC2SpotFleetRole 策略右侧的 x，然后选择分离。

竞价型实例无法缩减

2021 年 3 月 10 日AWS Batch推出了 AWSServiceRoleForBatch 服务相关角色。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 AmazonEC2SpotFleetTaggingRole 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将权限委托给AWS服务的角色。

主题

- 将 AmazonEC2SpotFleetTaggingRole 托管策略附加到在AWS 管理控制台中竞价型实例集角色
- 将 AmazonEC2SpotFleetTaggingRole 托管策略附加到在AWS CLI的竞价型实例集角色

将 AmazonEC2SpotFleetTaggingRole 托管策略附加到在AWS 管理控制台中竞价型实例集角色

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 AmazonEC2SpotFleetTaggingRole，然后选择附上策略。
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 AmazonEC2SpotFleetRole 策略右侧的 x，然后选择分离。

将 AmazonEC2SpotFleetTaggingRole 托管策略附加到在AWS CLI的竞价型实例集角色

示例命令假设 Amazon EC2 竞价型实例集角色名为 *AmazonEC2SpotFleetRole*。如果角色使用不同的名称，请调整命令以使其匹配。

将 AmazonEC2SpotFleetTaggingRole 托管策略附加到竞价型实例集角色上

1. 要将 AmazonEC2SpotFleetTaggingRole 托管 IAM policy 附加到*AmazonEC2SpotFleetRole* 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 AmazonEC2SpotFleetRole
```

2. 要将 AmazonEC2SpotFleetRole 托管 IAM policy 与 *AmazonEC2SpotFleetRole* 角色分离，请使用AWS CLI运行以下命令。

```
$ 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

对于在 containerOverrides 结构的 `memory` 和 `vcpus` 成员中指定的内存和 vCPU 覆盖（已传递给 SubmitJob），它们无法覆盖作业定义的 resourceRequirements 结构中指定的内存和 vCPU 要求。

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

使用AWS Batch API 更新所需的 vCPU (desiredvCpus) 设置时会出现以下错误消息。

```
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

请查看以下主题，了解在亚马逊 Elastic Kubernetes Service AWS Batch 上使用时可能遇到的常见问题的审查流程和潜在解决方案。

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：

- 1.34
- 1.33
- 1.32
- 1.31
- 1.30
- 1.29

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 CreateComputeEnvironment。
- 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 开始使用 Amazon EKS 上的 AWS Batch。

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。

Amazon EKS 作业上的AWS Batch停留在**RUNNABLE**状态

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 2 步：为 AWS Batch 准备您的 Amazon EKS 集群。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。

Amazon EKS 作业上的AWS Batch停留在**STARTING**状态

当容器组因来自 kubelet（pull、log、exec 和 attach）的任何长时间运行的请求而停滞在 ContainerCreating 的 PENDING 状态时，在容器组启动问题得到解决或作业被终止前，作业将停留在 STARTING 状态。在以下符合条件的情况下，AWS Batch 将代表您终止作业，否则必须使用 TerminateJob API 手动终止作业。

要验证作业停滞在 STARTING 状态的原因，请使用 教程：将正在运行的作业映射到容器组（pod）和节点 查找 podName 并描述该容器组：

```
% kubectl describe pod aws-batch.000c8190-87df-31e7-8819-176fe017a24a -n my-aws-batch-
namespace
Name:             aws-batch.000c8190-87df-31e7-8819-176fe017a24a
Namespace:        my-aws-batch-namespace
...
Containers:
  default:
...
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
...
Conditions:
  Type                        Status
  PodReadyToStartContainers   False
  Initialized                 True
```

```
  Ready                       False
  ContainersReady             False
  PodScheduled                True
...
Events:
  Type     Reason       Age    From     Message
  ----     ------       ----   ----     -------
  Warning  FailedMount  2m32s  kubelet  Unable to attach or mount volumes: ...
```

考虑将 EKS 集群配置为将控制面板日志发送到 CloudWatch Logs，以实现完全的可见性。

场景：持久卷声明连接或挂载失败

使用永久卷声明但卷连接或挂载失败的作业将会被终止。这可能是因作业定义配置不正确所致。有关更多信息，请参阅在 Amazon EKS 资源上创建单节点作业定义。

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 2 步：为 AWS Batch 准备您的 Amazon EKS 集群。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]
 events                                           []                 []
[list]
```

```
$ 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 2 步：为 AWS Batch 准备您的 Amazon EKS 集群。

资源：AWS Batch 服务配额

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 的计算环境 AWS Batch。	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
最大作业定义大小（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
最大消耗性资源数	5 万
最大服务环境数。有关更多信息，请参阅 服务环境。	50
每个作业队列的最大服务环境数量	1
SubmitServiceJob 请求的最大大小	30 KiB
最大作业服务请求有效载荷大小（对于 SubmitServiceJob API 操作）	10 KiB
每个 SubmitServiceJob 操作账户每秒 (TPS) 可处理的最大处理数	5
服务作业使用重试策略的最大尝试次数	10

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。

文档历史记录

下表描述了自AWS Batch初始发布以来对文档所做的重要更改。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
新增 default_x86_64 和 default_arm64 选项	允许的实例类型新增了 `default_x86_64` 和 `default_arm64` 。	2025 年 8 月 15 日
新增服务环境和服务作业	增加了用于将 AWS Batch 与 SageMaker AI 配合使用的服务环境和服务作业。	2025 年 7 月 30 日
新增 AWSServiceRoleForAWSBatchWithSagemaker 和 AWSBatchServiceRolePolicyForSageMaker	新增了 AWS 服务相关角色 `AWSServiceRoleForAWSBatchWithSagemaker` 和托管式策略 `AWSBatchServiceRolePolicyForSageMaker` ，以允许 AWS Batch 您代表您管理 SageMaker AI。	2025 年 7 月 30 日
增加对 EKS AL 2023 AMI 的支持	如何从 EKS AL2 升级到 EKS AL2023	2025 年 6 月 24 日
增加对 FireLens 和 ECS Exec 命令的支持	增加了对 FireLens 和 ECS Exec 命令的支持。	2025 年 4 月 15 日
增加对 AWS Batch 资源感知型调度功能的支持	为 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 和 AWS Fargate 增加对 AWS Batch 资源感知型调度功能的支持。	2025 年 2 月 27 日

更新了 AWS Batch 支持的 Amazon EKS 版本	更新了 AWS Batch 支持的 Amazon EKS 版本，以移除版本 1.22。	2024 年 3 月 11 日
更新了 AWS Batch 支持的 Amazon EKS 版本	更新了 AWS Batch 支持的 Amazon EKS 版本，以包含版本 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
增加对 AWS Batch 的多容器作业的支持	为 Amazon Elastic Container Service、Amazon Elastic Kubernetes Service 和 AWS Fargate 增加对 AWS Batch 的多容器作业的支持。	2024 年 2 月 28 日
更新了 AWS Batch 支持的 Amazon EKS 版本	更新了 AWS Batch 支持的 Amazon EKS 版本，以包含版本 1.28	2024 年 1 月 27 日
更新了BatchServiceRolePolicy 和 AWSBatchServiceRole	BatchServiceRolePolicy 进行了更新，以增加对描述竞价型实例集请求历史记录和 Amazon EC2 Auto Scaling 活动的支持。 AWSBatchServiceRole 进行了更新，以添加语句 ID、向 `ec2:DescribeSpotFleetRequestHistory` 和 `autoscaling:DescribeScalingActivities` 授予 AWS Batch 权限。	2023 年 12 月 5 日

AWS BatchAmazon EKS 上的	AWS Batch添加了对在 Amazon EKS 集群上运行作业的支持。	2022 年 10 月 25 日
针对AWS Batch的防止跨服务混淆代理	AWS Batch现在提供了一种解决混淆代理安全问题的方法，当一个实体（服务或账户）被另一个实体强迫执行操作时，就会出现该问题。	2022 年 6 月 6 日
接口 VPC 端点 (AWS PrivateLink)	添加了对配置由AWS PrivateLink提供支持的接口 VPC 端点的支持。这意味着可以在 VPC 和AWS Batch之间创建私有连接，而无需通过 NAT 实例、VPN 连接或Direct Connect进行访问。	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托管式策略更新 – 对现有策略的更新	AWS Batch更新了现有托管策略。	2021 年 12 月 6 日
公平份额调度	AWS Batch添加了对作业队列的计划策略的支持。	2021 年 11 月 9 日
Amazon EFS	AWS Batch添加了对将 Amazon EFS 文件系统添加到作业定义的支持。	2021 年 4 月 1 日
添加了服务相关角色	AWS Batch添加了 AWSServiceRoleForBatch 服务相关角色。	2021 年 3 月 10 日
AWS Fargate支持	AWS Batch添加了对在 Fargate 资源上运行任务的支持。	2020 年 12 月 3 日

为资源加标签	AWS Batch添加了对向计算环境、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添加了对向作业传递密文的支持。	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添加了对多个用于选择实例类型的策略的支持。	2019 年 10 月 16 日
EFA 支持	AWS Batch添加了对 Elastic Fabric Adapter (EFA) 设备的支持。	2019 年 8 月 2 日
GPU 计划	AWS Batch添加了 GPU 计划。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添加了对多节点并行作业的支持。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添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添加了对作业超时时间的支持。利用这项支持，可以为作业配置具体的超时时间，以便在某个作业运行超时时，AWS Batch就会终止作业。	2018 年 4 月 5 日

AWS Batch作业作为 EventBridge 的目标	AWS Batch作业可作为 EventBridge 的目标提供。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
针对的 CloudTrail 审计AWS Batch	CloudTrail 可审计对AWS Batch API 操作的调用。	2018 年 1 月 10 日
数组作业	AWS Batch添加了对数组作业的支持。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加标签功能	AWS Batch扩展了对加标签功能的支持。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 BatchEventBridge 事件流	AWS Batch为 EventBridge 添加事件流。可以使用AWS Batch事件流接收近乎实时的通知，了解提交到作业队列的作业的状态。	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通用版	引入了AWS Batch，作为在AWS Cloud上运行批处理计算工作负载的一种手段。	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。