# AWS Batch

## 用户指南

# AWS Batch: 用户指南

# Table of Contents

# 什么是 AWS Batch？

利用 AWS Batch，您可以在 AWS 云上运行批量计算工作负载。批量计算是开发人员、科学家和工程师用来访问大量计算资源的常见方法，并且 AWS Batch 将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 可让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，从而使您可以将精力放在分析结果和解决问题上。

# AWS Batch 组成部分

AWS Batch 是一种区域服务，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。

## 工作

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 Amazon EC2 实例上作为容器化应用程序运行 (使用您在任务定义中指定的参数)。任务可以按名称或按 ID 引用其他任务，并且可以依赖于其他任务的成功完成。有关更多信息，请参阅 工作 (p. 12)。

## 作业定义

任务定义指定任务如何运行；您可以把它看成是任务中的资源的蓝图。您可以为您的任务提供 IAM 角色，以便对其他 AWS 资源进行编程访问，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 Job Definitions (p. 29)。

## 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。您将一个或多个计算环境与一个任务队列相关联，并且可以为这些计算环境甚至跨任务队列本身分配优先级值。例如，您可以有一个高优先级队列用以提交时间敏感型任务，以及一个低优先级队列供可在计算资源较便宜时随时运行的任务使用。

## 计算环境

计算环境是一组用于运行任务的托管或非托管计算资源。托管计算环境能让您在多个详细级别指定所需的实例类型。您可以设置使用特定类型实例的计算环境，例如 `c4.2xlarge` 或 `m4.10xlarge`，或者只需指定您希望使用最新实例类型的计算环境。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 Spot 实例支付的金额占按需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 EC2 实例。您还可以管理自己的计算环境。在这种情况下，您负责在 AWS Batch 为您创建的 Amazon ECS 集群中设置和扩展实例。有关更多信息，请参阅计算环境 (p. 52)。

# 入门

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

AWS Batch 首次运行向导为您提供了创建计算环境和作业队列并提交示例 hello world 作业的选项。如果您具有要在 AWS Batch 中启动的 Docker 镜像，则可以使用此映像创建作业定义并改为将此定义提交到您的队列。有关更多信息，请参阅 AWS Batch 入门 (p. 8)。

# Setting Up with AWS Batch

如果您已注册 Amazon Web Services (AWS) 并已在使用 Amazon Elastic Compute Cloud (Amazon EC2) 或 Amazon Elastic Container Service (Amazon ECS)，您与使用 AWS Batch 已近在咫尺。这些服务的设置过程非常相似，因为 AWS Batch 在其计算环境中使用 Amazon ECS 容器实例。要对 AWS Batch 使用 AWS CLI，必须使用支持最新 AWS Batch 功能的 AWS CLI 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。如果您已完成以下任何步骤，可以跳过这些步骤并继续安装 AWS CLI。

1. Sign Up for AWS (p. 3)
2. Create an IAM User (p. 3)
3. Create IAM Roles for your Compute Environments and Container Instances (p. 5)
4. Create a Key Pair (p. 5)
5. Create a Virtual Private Cloud (p. 6)
6. Create a Security Group (p. 7)
7. Install the AWS CLI (p. 7)

## Sign Up for AWS

当您注册 AWS 时，您的 AWS 账户会自动注册所有服务，包括 Amazon EC2 和 AWS Batch。您只需为使用的服务付费。

如果您已有一个 AWS 账户，请跳到下一个任务。如果您还没有 AWS 账户，请使用以下步骤创建。

创建 AWS 账户

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

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

请记下您的 AWS 账号，因为在下一个任务中您会用到它。

## Create an IAM User

AWS 中的服务（例如 Amazon EC2 和 AWS Batch）要求您在访问时提供凭证，以便服务可以确定您是否有权限访问其资源。控制台要求您的密码。您可以为您的 AWS 账户创建访问密钥以访问命令行界面或 API。不过，我们不建议您使用 AWS 账户的证书访问 AWS，而建议您使用 AWS Identity and Access

Management (IAM)。创建 IAM 用户，然后将该用户添加到具有管理权限的 IAM 组或授予此用户管理权限。随后，您可以使用特殊 URL 和 IAM 用户的凭证访问 AWS。

如果您已注册 AWS 但尚未为自己创建一个 IAM 用户，则可以使用 IAM 控制台自行创建。

自行创建管理员用户并将该用户添加到管理员组（控制台）

1. 通过选择 根用户，然后输入您的 AWS 账户的电子邮件地址，以账户拥有者身份登录到 IAM 控制台。在下一页上，输入您的密码。

   Note

   强烈建议您遵守以下使用 **`Administrator`** IAM 用户的最佳实践，妥善保存根用户凭证。只在执行少数账户和服务管理任务时才作为根用户登录。
2. 在导航窗格中，选择用户，然后选择添加用户。
3. 对于 User name (用户名)，输入 **`Administrator`**。
4. 选中 AWS 管理控制台 访问 旁边的复选框。然后选择自定义密码，并在文本框中输入新密码。
5. （可选）默认情况下，AWS 要求新用户在首次登录时创建新密码。您可以清除 User must create a new password at next sign-in (用户必须在下次登录时创建新密码) 旁边的复选框以允许新用户在登录后重置其密码。
6. 选择下一步: 权限。
7. 在设置权限下，选择将用户添加到组。
8. 选择创建组。
9. 在 Create group (创建组) 对话框中，对于 Group name (组名称)，输入 **`Administrators`**。
10. 选择 Filter policies (筛选策略)，然后选择 AWS managed-job function (AWS 托管的工作职能) 以筛选表内容。
11. 在策略列表中，选中 AdministratorAccess 的复选框。然后选择 Create group (创建组)。

    Note

    您必须先激活 IAM 用户和角色对账单的访问权限，然后才能使用 `AdministratorAccess` 权限访问 AWS Billing and Cost Management 控制台。为此，请按照“向账单控制台委派访问权限”教程第 1 步中的说明进行操作。
12. 返回到组列表中，选中您的新组所对应的复选框。如有必要，选择 Refresh 以在列表中查看该组。
13. 选择下一步: 标签。
14. （可选）通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 IAM 用户指南 中的标记 IAM 实体。
15. 选择 Next: Review (下一步: 审核) 以查看要添加到新用户的组成员资格的列表。如果您已准备好继续，请选择 Create user。

您可使用此相同的流程创建更多的组和用户，并允许您的用户访问 AWS 账户资源。要了解有关使用策略限制用户对特定 AWS 资源的权限的信息，请参阅访问管理和示例策略。

登录 IAM 用户，注销AWS控制台，然后使用以下URL， your_aws_account_id 您的AWS帐号没有连字符（例如，如果您的AWS账号是 `1234-5678-9012`，您的AWS帐户ID是 `123456789012`):

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

输入您刚创建的 IAM 用户名和密码。登录后，导航栏会显示"your_user_name @ your_aws_account_id“”。

如果您不希望您的登录页面 URL 包含 AWS 账户 ID，可以创建账户别名。从 IAM 控制面板中，选择 Create Account Alias (创建账户别名)，然后输入一个别名，例如您的公司名称。要在创建账户别名后登录，请使用以下 URL：

```
https://your_account_alias.signin.aws.amazon.com/console/
```

要为您的账户验证 IAM 用户的登录链接，请打开 IAM 控制台并在控制面板的 IAM users sign-in link (IAM 用户登录链接) 下进行检查。

有关 IAM 的更多信息，请参阅 AWS Identity and Access Management 用户指南。

# Create IAM Roles for your Compute Environments and Container Instances

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。您必须创建可将这些凭证提供给计算环境和容器实例的 IAM 角色，然后将该角色与计算环境关联。

> Note
>
> 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，则可以继续下一个部分。如果您计划改用 AWS CLI，请先完成 AWS Batch Service IAM Role (p. 83)和 Amazon ECS Instance Role (p. 86)中的过程，然后再创建您的第一个计算环境。

# Create a Key Pair

AWS 使用公共密钥密码术来保护您的实例的登录信息。Linux 实例 (例如，AWS Batch 计算环境容器实例) 没有用于 SSH 访问的密码；您使用密钥对安全登录您的实例。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，如果您计划在多个区域启动实例，则需要在每个区域中创建密钥对。有关区域的更多信息，请参阅https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.htmlAmazon EC2 用户指南（适用于 Linux 实例）中的区域和可用区。

创建密钥对

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中，选择密钥对区域。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）区域中启动实例，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。选择一个容易记住的名称，例如，您的 IAM 用户名，后跟 `-key-pair` 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。基本文件名称是指定为密钥对名称的名称，文件名扩展名为 `.pem`。将私钥文件保存在安全的地方。

   > Important
   >
   > 这是您保存私有密钥文件的唯一机会。当您启动实例时，您将需要提供密钥对的名称；当您每次连接到实例时，您将需要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，以确保只有您可以读取它。

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

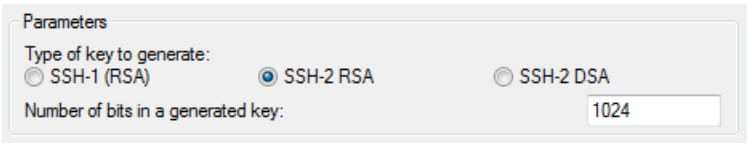
有关更多信息，请参阅 Amazon EC2 中的 Amazon EC2 用户指南（适用于 Linux 实例）密钥对。

使用密钥对连接到实例

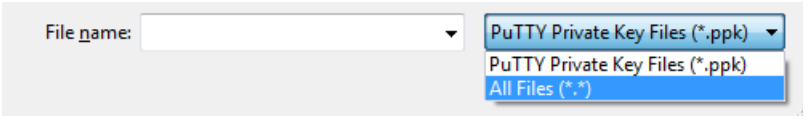
要从运行 Mac 或 Linux 的计算机连接到 Linux 实例，需要使用 -i 选项对 SSH 客户端指定 .pem 文件和私有密钥的路径。若要从运行 Windows 的计算机连接到 Linux 实例，可以使用 MindTerm 或 PuTTY。如果您计划使用 PuTTY，则需要安装它并遵循以下步骤将 .pem 文件转换为 .ppk 文件。

（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。请务必安装整个套件。
2. 启动 PuTTYgen（例如，在开始菜单中，依次单击所有程序、PuTTY 和 PuTTYgen）。
3. 在 Type of key to generate 下，选择 SSH-2 RSA。




4. 选择 Load。默认情况下， PuTTYgen 仅显示扩展名的文件 .ppk。查找您的 .pem 文件，选择显示所有类型文件的选项。




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 Save private key (保存私有密钥)。PuTTYgen 会显示一条警告，提示将在未提供口令的情况下保存密钥。选择是。
7. 为密钥指定密钥对所用的相同名称。PuTTY 会自动添加 .ppk 文件扩展名。

# Create a Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) 允许您在已定义的虚拟网络内启动 AWS 资源。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，即 Create a Security Group (p. 7)。要确定您是否有默认VPC，请参阅 AmazonEC2控制台中支持的平台 在 Amazon EC2 用户指南（适用于 Linux 实例）。否则，您可以使用以下步骤在帐户中创建非默认VPC。

> **Important**
>
> 如果您的账户在某个区域中支持 EC2-Classic，则您在该区域没有默认 VPC。

创建非默认 VPC

1. 打开 Amazon VPC 控制台 https://console.aws.amazon.com/vpc/。
2. 从导航栏中，为 VPC 选择区域。VPC 特定于某一区域，因此您应选择已创建密钥对的区域。
3. 在 VPC 控制面板上，选择 Start VPC Wizard。
4. 在 步骤1: 选择VPC配置 页面，确保 具有单个公共子网的VPC 选择并选择 选择.
5. 在 步骤2: 具有单个公共子网的VPC 页面，输入VPC的友好名称 VPC名称. 保留其他默认配置设置，然后选择 Create VPC。在确认页面上，请选择 OK。

有关 Amazon VPC，参见 什么是AmazonVPC？ 在 Amazon VPC 用户指南.

# Create a Security Group

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。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 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.htmlAmazon EC2 用户指南（适用于 Linux 实例）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 Internet 服务提供商 (ISP) 连接或者在不使用静态 IP 的情况下从防火墙后面连接，则您需要找出客户端计算机使用的 IP 地址范围。

创建具有最小特权的安全组

1. 打开 Amazon EC2 控制台 https://console.aws.amazon.com/ec2/。
2. 从导航栏中选择安全组的区域。安全组特定于某一区域，因此您应选择已创建密钥对的区域。
3. 在导航窗格中，选择 Security Groups，然后选择 Create Security Group。
4. 输入新安全组的名称和描述。选择一个容易记住的名称，例如，您的 IAM 用户名称，后跟 _SG_ 加区域名称。例如， meSG_useast1.
5. 在 VPC 列表中，确保选择您的默认 VPC；它标有星号 (*)。

   > Note
   >
   > 如果您的账户支持 EC2-Classic，请选择您在上一个任务中创建的 VPC。
6. AWS Batch 容器实例不需要打开任何入站端口。但您可能需要添加 SSH 规则，以便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：

   - Choose Add Rule. For Type, choose HTTP. For Source, choose Anywhere (`0.0.0.0/0`).
   - Choose Add Rule. For Type, choose SSH. For Source, ensure that Custom IP is selected, and specify the public IP address of your computer or network in CIDR notation. To specify an individual IP address in CIDR notation, add the routing prefix `/32`. For example, if your IP address is `203.0.113.25`, specify `203.0.113.25/32`. If your company allocates addresses from a range, specify the entire range, such as `203.0.113.0/24`.

     > Note
     >
     > For security reasons, we don't recommend that you allow SSH access from all IP addresses (`0.0.0.0/0`) to your instance, except for testing purposes and only for a short time.

# Install the AWS CLI

要对 AWS Batch 使用 AWS CLI，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 https://docs.aws.amazon.com/cli/latest/userguide/installing.htmlAWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。

# AWS Batch 入门

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

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

Important

在开始之前，请确保您已完成Setting Up with AWS Batch (p. 3)中的步骤，并确保您的 AWS 用户具有所需的权限 (管理员用户无需担心权限问题)。有关更多信息，请参阅 IAM 用户指南 中的创建您的第一个 IAM 管理员用户和组。

# 步骤 1：定义作业

在此部分，您选择定义您的作业定义或在没有作业定义的情况下继续创建计算环境和作业队列。

### 配置作业选项

1. 在 https://console.aws.amazon.com/batch/home#/wizard 中打开 AWS Batch 控制台首次运行向导。
2. 要创建 AWS Batch 任务定义、计算环境和任务队列，然后提交您的任务，请选择使用 Amazon EC2。要仅创建计算环境和作业队列而不提交作业，请选择 No job submission。
3. 如果您选择创建作业定义，请完成首次运行向导下面的四个部分：Job run-time、Environment、Parameters 和 Environment variables，然后选择 Next。如果您不创建作业定义，请选择 Next 并转到步骤 2：配置计算环境和作业队列 (p. 9)。

### 指定作业运行时间

1. 如果您要创建新的作业定义，对于 Job definition name，请指定作业定义的名称。
2. （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。有关此功能的更多信息（包括配置先决条件），请参阅 Amazon Elastic Container Service Developer Guide 中的适用于任务的 IAM 角色。

   Note

   此处仅显示具有 Amazon Elastic Container Service 任务角色信任关系的角色。有关为 AWS Batch 任务创建 IAM 角色的更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的为任务创建 IAM 角色和策略。
3. 对于 Container image，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。您也可以使用 *repository-url*/*image*:*tag* 指定其他存储库。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。此参数将映射到 Docker Remote API 的创建容器部分中的 `Image` 以及 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 存储库中的映像使用完整的 `registry/repository:tag` 命名约定。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。

为环境指定资源

1. 对于 Command，指定要传递到容器的命令。此参数将映射到 Docker Remote API 的创建容器部分中的 `Cmd` 以及 docker run 的 `COMMAND` 参数。有关 Docker `CMD` 参数的更多信息，请转到 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 (p. 37)。
2. 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。
3. 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。
4. 对于 Job attempts，指定尝试作业的最大次数 (在尝试失败的情况下)。有关更多信息，请参阅自动作业重试 (p. 16)。

参数

您可以选择在命令中指定参数替代默认值和占位符。有关更多信息，请参阅 Parameters (p. 37)。

1. 对于 Key，指定参数的键。
2. 对于 Value，指定参数的值。

指定环境变量

您可以选择性地指定要传递到您的作业容器的环境变量。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。

1. 对于 Key，指定环境变量的键。
2. 对于 Value，指定环境变量的值。

# 步骤 2：配置计算环境和作业队列

计算环境是一种引用计算资源（Amazon EC2 实例）的方式：告知 AWS Batch 如何配置和自动启动实例的设置和限制。您将作业提交到一个作业队列，该队列将一直存储作业，直至 AWS Batch 计划程序在计算环境中的计算资源上运行作业。

Note

目前，您只能在首次运行向导中创建托管计算环境。要创建非托管计算环境，请参阅创建计算环境 (p. 63)。

### 配置您的计算环境类型

1. 对于 Compute environment name，为您的计算环境指定唯一名称。
2. 对于服务角色，选择创建新角色或使用现有角色，后者允许 AWS Batch 服务代表您调用所需的 AWS API。有关更多信息，请参阅AWS Batch Service IAM Role (p. 83)。如果您选择创建新角色，则将为您创建所需的角色 (`AWSBatchServiceRole`)。
3. 对于 EC2 实例角色，选择创建新角色或使用现有角色；通过角色，为计算环境创建的 Amazon ECS 容器实例可代表您调用所需的 AWS API。有关更多信息，请参阅Amazon ECS Instance Role (p. 86)。如果您选择创建新角色，则将为您创建所需的角色 (`ecsInstanceRole`)。

### 配置实例

1. 对于预配置模型，选择按需以启动 Amazon EC2 按需实例，或选择 Spot 以使用 Amazon EC2 Spot 实例。
2. 在选择使用 Amazon EC2 Spot 实例的情况下：

   a. 对于 Maximum bid price，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 Spot 实例价格必须达到的最大百分比。例如，如果出价百分比为 20%，则 Spot 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。

   b. 对于 Spot 队组角色，选择创建新角色或使用要应用于您的 Spot 计算环境的现有 Amazon EC2 Spot 队组 IAM 角色。如果您选择创建新角色，则将为您创建所需的角色 (`aws-ec2-spot-fleet-role`)。有关更多信息，请参阅Amazon EC2 Spot Fleet Role (p. 86)。
3. 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启动这些系列中的任何实例类型（例如，`c5`、`c5n` 或 `p3`），也可以指定系列中的特定大小（例如，`c5.8xlarge`）。请注意，裸机实例类型不包括在实例系列中（例如，`c5` 不包括 `c5.metal`）。 您也可以选择 `optimal` 以（从最新的 C、M 和 R 实例系列中）动态选取符合作业队列要求的实例类型。

   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。
4. 对于 Minimum vCPUs (最小 vCPU 数)，选择您的计算环境应保留的 EC2 vCPU 的最小数目，而无论作业队列需求如何。
5. 对于 Desired vCPUs，请选择您的计算环境在启动时应使用的 EC2 vCPU 数量。当任务队列需求增大时，AWS Batch 会增加计算环境中所需的 vCPU 数并添加 EC2 实例（最多可达最大 vCPU 数）；当需求减少时，AWS Batch 会减少计算环境中所需的 vCPU 数并删除实例（减少至最小 vCPU 数）。
6. 对于 Maximum vCPUs，选择您的计算环境可以向外扩展到的 EC2 vCPU 的最大数目，而无论作业队列需求如何。

### 设置您的网络

计算资源将在您在此处指定的 VPC 和子网中启动。这使您能够控制 AWS Batch 计算资源的网络隔离。

Important

计算资源需要访问权限以便与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 终端节点或通过具有公有 IP 地址的计算资源完成。
有关接口 VPC 终端节点的更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink)。
如果您没有配置接口 VPC 终端节点并且您的计算资源没有公有 IP 地址，则它们必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关。有关更多信息，请参阅教程：为您的计算环境创建带有公有和私有子网的 VPC (p. 100)。

1. 对于 VPC Id，选择在其中启动实例的 VPC。
2. 对于 Subnets，选择选定 VPC 中应托管实例的子网。默认情况下，将选择选定 VPC 中的所有子网。

3. 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。

### 为您的实例添加标签

您可以选择将键/值对标签应用于计算环境中启动的实例。例如，您可以指定 `"Name": "AWS Batch Instance - C4OnDemand"` 作为标签，以便计算环境中的每个实例均具有此名称（这对于在 Amazon EC2 控制台中识别您的 AWS Batch 实例很有用）。默认情况下，计算环境名称用于为您的实例添加标签。

1. 对于 Key，指定标签的键。
2. 对于 Value，指定标签的值。

### 设置作业队列

您将作业提交到一个作业队列，该队列将一直存储作业，直至 AWS Batch 计划程序在计算环境中的计算资源上运行作业。

- 对于 Job queue name，请为您的作业队列选择唯一的名称。

### 查看和创建

Connected compute environments for this job queue 部分显示您的新计算环境与您的新作业队列关联及其顺序。稍后，您可以将其他计算环境与作业队列关联。作业计划程序使用计算环境顺序来确定哪些计算环境应执行给定作业。计算环境必须先处于 `VALID` 状态，然后您才能将其与作业队列关联。您最多可以将三个计算环境与一个作业队列关联。

- 检查计算环境和作业队列配置，并选择 Create 以创建计算环境。

# 工作

作业是由 AWS Batch 执行的工作单位。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执行。

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 Job Definition Parameters (p. 36)。

您可以提交大量独立的简单作业。

主题



## 提交作业

在注册作业定义后，您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. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Jobs 和 Submit job。
4. 对于 Job name，请为您的队列选择唯一的名称。
5. 对于 Job definition，为作业选择之前创建的作业定义。有关更多信息，请参阅Creating a Job Definition (p. 29)。
6. 对于 Job queue，选择之前创建的作业队列。有关更多信息，请参阅创建作业队列 (p. 48)。
7. 对于 Job type，为单个作业选择 Single，或选择 Array 以提交数组作业。有关更多信息，请参阅 Array Jobs (p. 17)。此选项不适用于多节点并行作业。
8. (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。
9. (可选) 声明任何作业依赖项。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 (p. 16)。

    a. 对于 Job depends on，输入必须在此作业启动前完成的任何作业的作业 ID。

    b. (仅限数组作业) 对于 N-To-N job dependencies，指定任何数组作业的一个或多个作业 ID (此作业的每个子作业索引均依赖于依赖项的相应子索引作业)。例如，`JobB:1` 依赖于 `JobA:1`，依此类推。

    c. (仅限数组作业) 选择 Run children sequentially 来为当前数组作业创建 `SEQUENTIAL` 依赖项。这可确保每个子索引作业等待其之前的作业完成。例如，`JobA:1` 依赖于 `JobA:0`，依此类推。

10. 对于 Job attempts，指定尝试作业的最大次数 (在尝试失败的情况下)。有关更多信息，请参阅自动作业重试 (p. 16)。
11. （可选）对于 Execution timeout (执行超时)，指定允许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 (p. 16)。
12. （可选）在 Parameters (参数) 部分中，您可以指定参数替代默认值和占位符，以便在您的作业容器启动时所运行的命令中使用。有关更多信息，请参阅Parameters (p. 37)。

    a. 选择 Add parameter (添加参数)。
    b. 对于 Key，指定参数的键。
    c. 对于 Value，指定参数的值。
13. 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。
14. 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。您必须为任务指定至少 4 MiB 内存。
15. （可选）对于 Number of GPU ，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。
16. 对于 Command，指定要传递到容器的命令。对于简单的命令，您可以在 Space delimited 选项卡上键入命令，就像在命令提示符中键入命令一样。确保 JSON 结果 (该结果将传递到 Docker 守护程序) 正确无误。对于较复杂的命令 (例如，带有特殊字符)，您可以切换到 JSON 选项卡，然后在该选项卡中输入等效字符串数组。

    此参数将映射到 Docker Remote API 的创建容器部分中的 `Cmd` 以及 docker run 的 `COMMAND` 参数。有关 Docker `CMD` 参数的更多信息，请转到 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 (p. 37)。
17. (可选) 您可以指定要传递到您的作业容器的环境变量。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。

    a. 选择 Add environment variable (添加环境变量)。
    b. 对于 Key，指定环境变量的键。

       > **Note**
       >
       > 环境变量不能以`AWS_BATCH` 开头；此命名约定是为 AWS Batch 服务所设置的变量保留的。

    c. 对于 Value，指定环境变量的值。
18. 选择 Submit job。

    > **Note**
    >
    > CloudWatch Logs 中提供 `RUNNING`、`SUCCEEDED` 和 `FAILED` 任务的日志；日志组是 `/aws/batch/job`，日志流名称格式为 *`jobDefinitionName`*`/default/`*`ecs_task_id`*（此格式在日后可能会更改）。
    > 在任务到达 `RUNNING` 状态后，您可以使用 DescribeJobs API 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 CloudWatch Logs User Guide 中的查看发送到 CloudWatch Logs 的日志数据。默认情况下，这些日志设置为永不过期，但您可以修改保留期。有关更多信息，请参阅 中的更改 中的日志数据保留期。

# 作业状态

当您将一个任务提交到 AWS Batch 任务队列时，该任务将进入 SUBMITTED 状态。随后，它将经历以下状态，直至成功（退出并返回代码 0）或失败（退出并返回非零代码）。AWS Batch 任务可具有以下状态：

SUBMITTED

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

PENDING

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

RUNNABLE

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

Note

如果您的任务未进行到 STARTING，请参阅故障排除部分中的 作业在 RUNNABLE 状态卡住 (p. 114)。

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。

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 (p. 16)。

Note

CloudWatch Logs 中提供 RUNNING 任务的日志，日志组是 /aws/batch/job，日志流名称格式为 *jobDefinitionName*/default/*ecs_task_id*（此格式在日后可能会更改）。在任务到达 RUNNING 状态后，您可以使用 DescribeJobs API 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 CloudWatch Logs User Guide 中的查看发送到 CloudWatch Logs 的日志数据。默认情况下，这些日志设置为永不过期，但您可以修改保留期。有关更多信息，请参阅 中的更改 中的日志数据保留期。

SUCCEEDED

作业已成功完成，并返回退出代码 0。SUCCEEDED 任务的任务状态在 AWS Batch 中保留 24 小时。

Note

CloudWatch Logs 中提供 SUCCEEDED 任务的日志，日志组是 /aws/batch/job，日志流名称格式为 *jobDefinitionName*/default/*ecs_task_id*（此格式在日后可能会更改）。在任务到达 RUNNING 状态后，您可以使用 DescribeJobs API 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 CloudWatch Logs User Guide 中的查看发送到 CloudWatch Logs 的日志数据。默认情况下，这些日志设置为永不过期，但您可以修改保留期。有关更多信息，请参阅 中的更改 中的日志数据保留期。

`FAILED`

在执行所有可用尝试后，作业失败。`FAILED` 任务的任务状态在 AWS Batch 中保留 24 小时。

Note

CloudWatch Logs 中提供 `FAILED` 任务的日志，日志组是 `/aws/batch/job`，日志流名称格式为 *`jobDefinitionName`*`/default/`*`ecs_task_id`*（此格式在日后可能会更改）。在任务到达 `RUNNING` 状态后，您可以使用 DescribeJobs API 操作以编程方式检索其日志流。有关更多信息，请参阅 Amazon CloudWatch Logs User Guide 中的查看发送到 CloudWatch Logs 的日志数据。默认情况下，这些日志设置为永不过期，但您可以修改保留期。有关更多信息，请参阅 中的更改 中的日志数据保留期。

# AWS Batch 作业环境变量

AWS Batch 自动在容器作业中设置特定环境变量。这些环境变量为作业内的容器提供了自检，并且您可以在应用程序的逻辑中使用这些变量的值。由 AWS Batch 设置的所有变量都以前缀 `AWS_BATCH_` 开头。这是一个受保护的环境变量前缀，您不能在作业定义或覆盖中将此前缀用于您自己的变量。

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 Tutorial: Using the Array Job Index to Control Job Differentiation (p. 21)。

`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 (p. 16)。

`AWS_BATCH_JOB_ID`

此变量设置为 AWS Batch 作业 ID。

`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（它在主节点上不存在）。此变量设置为作业的主节点的私有 IPv4 地址。您的子节点的应用程序代码可以使用此地址与主节点进行通信。

`AWS_BATCH_JOB_NODE_INDEX`

此变量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`

此变量设置为您的作业已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。可能的失败情况包括：

- 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 服务出错或中断

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允许您指定一个具有 1 至 10 次尝试的重试策略。有关更多信息，请参阅 Retry Strategy (p. 44)。

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)。

如果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，则会再次将作业置于 `RUNNABLE` 状态。有关更多信息，请参阅作业状态 (p. 14)。

> 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 Creating a Job Definition (p. 29) 和 提交作业 (p. 12)。

# 作业依赖项

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

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

对于数组作业，您可以指定 `SEQUENTIAL` 类型依赖项，而无需指定作业 ID，以便每个子数组作业按顺序完成 (从索引 0 开始)。您也可以使用作业 ID 指定 `N_TO_N` 类型依赖项。这样一来，此作业的每个子索引必须等待每个依赖项的相应子索引完成后才能开始。有关更多信息，请参阅Array Jobs (p. 17)。

要提交具有依赖项的 AWS Batch 任务，请参阅提交作业 (p. 12)。

# 作业超时

您可以为任务配置超时时间，以便在某个任务运行的时间超过超时时间时让 AWS Batch 终止该任务。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。

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。当此秒数已通过以下任务尝试的 `startedAt` 时间戳时，AWS Batch 会终止该任务。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。

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。

对于数组任务，子任务与父任务具有相同的超时配置。

有关提交带超时配置的 AWS Batch 作业的信息，请参阅提交作业 (p. 12)。

# Array Jobs

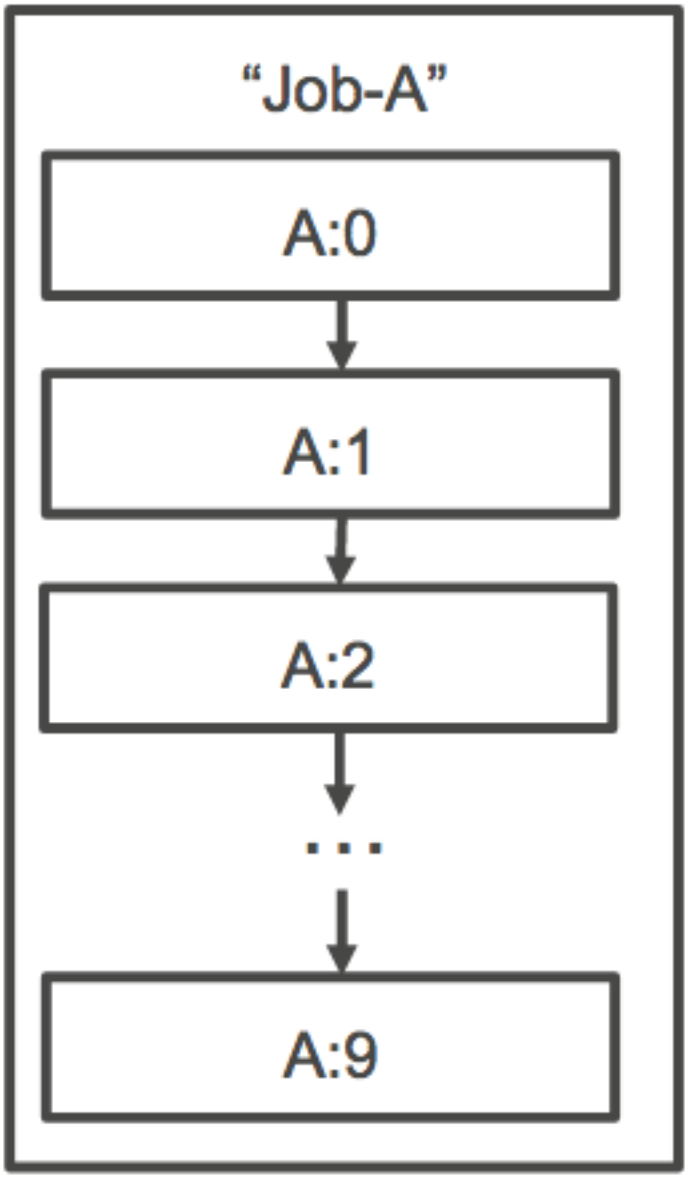
数组作业是共享通用参数 (如作业定义、vCPU 和内存) 的作业。它会以一系列相关但独立的基本作业的形式运行，这些作业可能跨多个主机分布，而且可能同时运行。数组作业是执行高度并行作业 (如 Monte Carlo 模拟、参数扫描或大型渲染作业) 的最高效方式。

像提交常规作业一样提交 AWS Batch 数组作业。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。这将允许您使用单个查询提交大型工作负载。

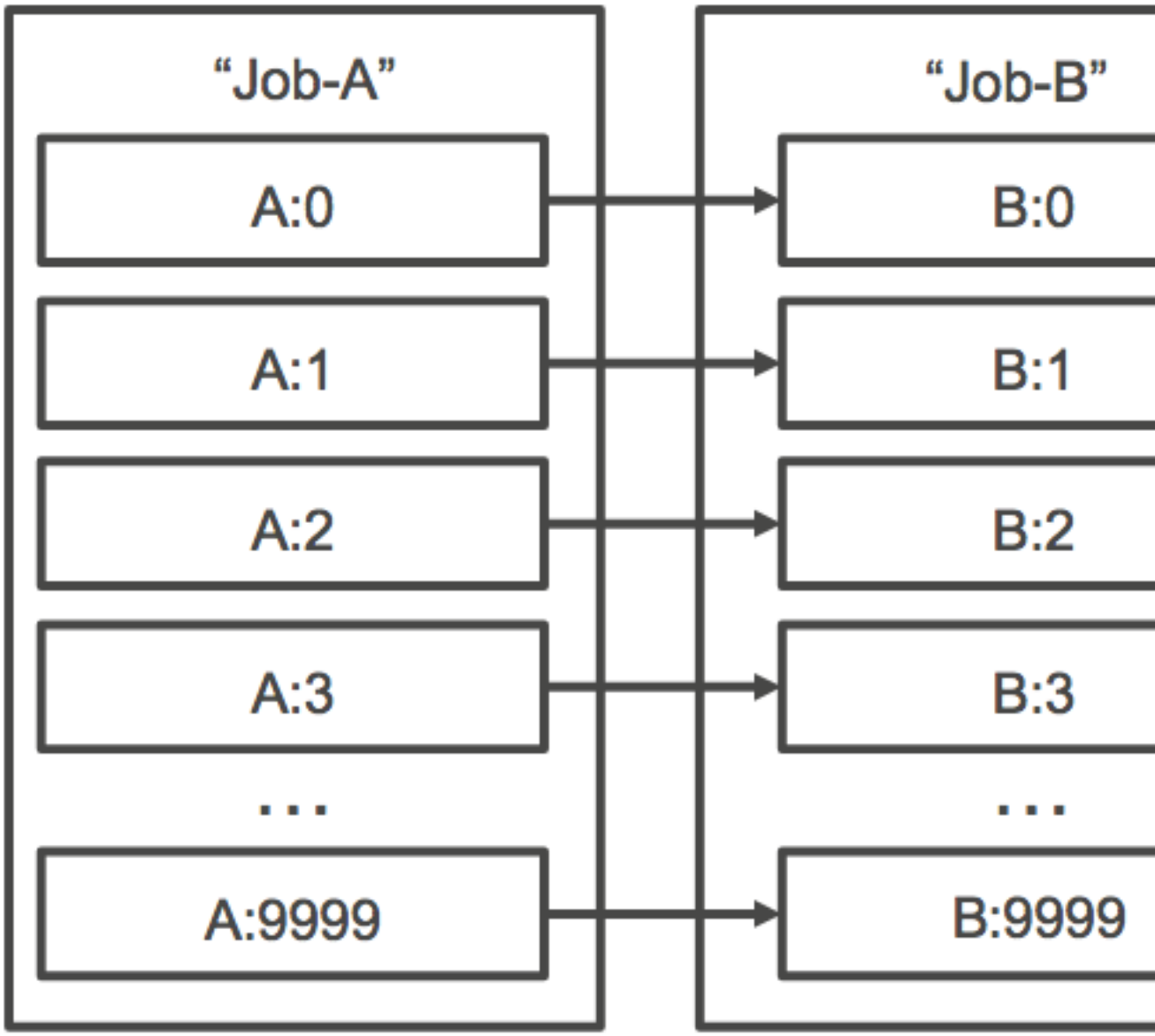
提交数组作业时，父数组作业会获取一般 AWS Batch 作业 ID。每个子作业都具有相同的基本 ID，但子作业的数组索引将附加到父 ID 的末尾，如数组的首个子作业的 *example_job_ID*`:0`。

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 Tutorial: Using the Array Job Index to Control Job Differentiation (p. 21)。)

对于阵列作业依赖关系，可以为依赖关系指定一个类型，例如 `SEQUENTIAL` 或 `N_TO_N`。您可以指定 `SEQUENTIAL` 键入依赖关系（不指定作业ID），以便每个子阵列作业顺序完成，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。




您也可以为数组作业指定具有作业 ID 的 `N_TO_N` 类型依赖项，以便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/终止)，则父作业也会失败。

# Example Array Job Workflow

AWS Batch 客户的常见工作流是运行先决条件设置任务，针对大量输入任务运行一系列命令，然后以聚合结果并将摘要数据写入到 Amazon S3、DynamoDB、Amazon Redshift 或 Aurora 的任务结束。

例如：。

- `JobA`: A standard, non-array job that performs a quick listing and metadata validation of objects in an Amazon S3 bucket, `BucketA`. The SubmitJob JSON syntax is shown below.

```
{
    "jobName": "JobA",
```

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

- `JobB`: An array job with 10,000 copies that is dependent upon `JobA`, that runs CPU-intensive commands against each object in `BucketA` and uploads results to `BucketB`. The SubmitJob JSON syntax is shown below.

```
{
    "jobName": "JobB",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobB-CPU-Intensive-Processing:1",
    "containerOverrides": {
        "vcpus": 32,
        "memory": 4096
   }
    "arrayProperties": {
        "size": 10000
    },
    "dependsOn": [
        {
            "jobId": "JobA_job_ID"
  }
    ]
}
```

- `JobC`: Another 10,000 copy array job that is dependent upon `JobB` with an `N_TO_N` dependency model, that runs memory-intensive commands against each item in `BucketB`, writes metadata to DynamoDB, and uploads the resulting output to `BucketC`. The SubmitJob JSON syntax is shown below.

```
{
    "jobName": "JobC",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobC-Memory-Intensive-Processing:1",
    "containerOverrides": {
        "vcpus": 1,
        "memory": 32768
   }
    "arrayProperties": {
        "size": 10000
    },
    "dependsOn": [
        {
            "jobId": "JobB_job_ID",
            "type": "N_TO_N"
        }
    ]
}
```

- `JobD`: An array job that performs 10 validation steps that each need to query DynamoDB and may interact with any of the above Amazon S3 buckets. Each of the steps in `JobD` run the same command, but the behavior is different based on the value of the `AWS_BATCH_JOB_ARRAY_INDEX` environment variable within the job's container. These validation steps run sequentially (for example, `JobD:0`, then `JobD:1`, and so on. The SubmitJob JSON syntax is shown below.

```
{
    "jobName": "JobD",
    "jobQueue": "ProdQueue",
    "jobDefinition": "JobD-Sequential-Validation:1",
    "containerOverrides": {
        "vcpus": 1,
        "memory": 32768
```

```
    }
    "arrayProperties": {
        "size": 10
    },
    "dependsOn": [
        {
            "jobId": "JobC_job_ID"
        },
        {
            "type": "SEQUENTIAL"
        },

    ]
}
```

- `JobE`: A final, non-array job that performs some simple cleanup operations and sends an Amazon SNS notification with a message that the pipeline has completed and a link to the output URL. The SubmitJob JSON syntax is shown below.

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

# Tutorial: Using the Array Job Index to Control Job Differentiation

本教程介绍如何使用 `AWS_BATCH_JOB_ARRAY_INDEX` 环境变量（所分配的每个子作业）来区分子作业。该示例的工作原理是使用子作业的索引数量，以读取文件中的特定行并用作业的容器内的命令替换与该行号关联的参数。最终，您可以拥有多个运行同一 Docker 映像和命令参数的 AWS Batch 作业，但结果由于数组作业索引被用作修饰符而不同。

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 0 开始，但行号从 1 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，您应注册使用新容器映像的作业定义，提交具有该作业定义的 AWS Batch 数组作业并查看结果。

## Prerequisites

本教程包含以下先决条件：

- An AWS Batch compute environment. For more information, see 创建计算环境 (p. 63).
- An AWS Batch job queue and associated compute environment. For more information, see 创建作业队列 (p. 48).
- The AWS CLI installed on your local system. For more information, see Installing the AWS Command Line Interface in the AWS Command Line Interface 用户指南.

- Docker installed on your local system. For more information, see About Docker CE in the Docker documentation.

## Step 1: Build a Container Image

尽管您可以在命令参数内的作业定义中使用 AWS_BATCH_JOB_ARRAY_INDEX，但创建在入口点脚本中使用变量的容器映像更方便、更高效。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
```

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.
```

## Step 2: Push Your Image to 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。本示例使用了 Amazon ECR，但您也可以选择使用其他注册表，例如 DockerHub。

1. 创建用于存储您的容器映像的 Amazon ECR 映像存储库。为简便起见，本示例使用了 AWS CLI，但您也可以使用 AWS 管理控制台。有关更多信息，请参阅 https://docs.aws.amazon.com/AmazonECR/latest/userguide/repository-create.htmlAmazon 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 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 https://docs.aws.amazon.com/AmazonECR/latest/userguide/Registries.html#registry_authAmazon 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
   ```

## Step 3: Create and Register a Job Definition

既然您的 Docker 映像位于映像注册表中，您可以在 AWS Batch 作业定义中指定它并在稍后使用它来运行数组作业。为简便起见，本示例使用了 AWS CLI，但您也可以使用 AWS 管理控制台。有关更多信息，请参阅 Creating a Job Definition (p. 29)。）

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",
       "vcpus": 1,
       "memory": 250
     }
   }
   ```

2. 使用 AWS Batch 注册作业定义：

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

## Step 4: Submit an AWS Batch Array Job

在注册作业定义后，您可以提交使用新容器映像的 AWS Batch 数组作业。

### 提交 AWS Batch 数组作业

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

   本示例采用由 AWS Batch 首次运行向导创建的默认作业队列名称。如果您的作业队列名称不同，请将 `first-run-job-queue` 名称替换为您的作业队列名称。

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

2. 将作业提交到 AWS Batch 作业队列。记下在输出中返回的作业 ID。

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

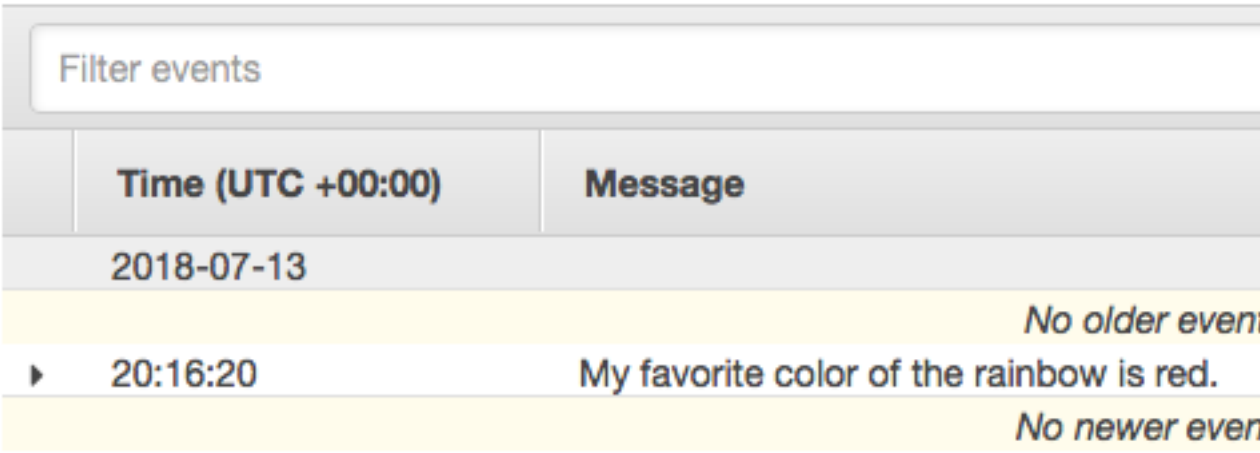
3. 描述作业的状态并等待作业移动到 `SUCCEEDED`。

## Step 5: View Your Array Job Logs

在任务达到 `SUCCEEDED` 状态后，您可以从任务的容器查看 CloudWatch Logs。

### 在 CloudWatch Logs 中查看作业的日志

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




7. 查看其他子作业的日志。每个作业将分别返回彩虹的一种颜色。

# 多节点并行作业

利用多节点并行作业，您能够跨多个 Amazon EC2 实例运行单个作业。借助 AWS Batch 多节点并行作业，您可以运行大规模、紧密耦合的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 多节点并行作业与支持基于 IP 的节点间通信的任何框架均可兼容，例如 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。如果主节点退出，则作业将视为已完成，并且子节点将停止。有关更多信息，请参阅节点组 (p. 26)。

多节点并行作业的节点为单租户，这意味着每个 Amazon EC2 实例上只有单个作业容器运行。

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。节点使用 *#N* 表示法进行寻址。例如，要访问作业第二个节点的详细信息，需要使用 AWS Batch DescribeJobs API 操作描述 *aws_batch_job_id*#2。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，则主节点故障将触发另一次尝试，子节点故障则不会。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 AWS Batch 上运行多节点并行作业，应用程序代码必须包含进行分布式通信所需的框架和库。

## 环境变量

在运行时，除了所有 AWS Batch 作业收到的标准环境变量之外，每个节点还将配置特定于多节点并行作业的以下环境变量：

`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（它在主节点上不存在）。此变量设置为作业的主节点的私有 IPv4 地址。您的子节点的应用程序代码可以使用此地址与主节点进行通信。

`AWS_BATCH_JOB_NODE_INDEX`

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

`AWS_BATCH_JOB_NUM_NODES`

此变量设置为您为多节点并行作业请求的节点数。

# 节点组

节点组即共享相同容器属性的一组完全相同的作业节点。您可以通过 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，您可以提交以下作业，其中主节点需要单个 `c4.xlarge` 实例，子节点需要五个 `c4.xlarge` 实例。每个不同的节点组可以为每个作业指定不同的容器映像或要运行的命令。

或者，作业中的所有节点可以使用单个节点组，应用程序代码可以通过比较 `AWS_BATCH_JOB_MAIN_NODE_INDEX` 环境变量与其自己的 `AWS_BATCH_JOB_NODE_INDEX` 值来区分节点角色（主节点与子节点）。单个作业中最多可具有 1000 个节点。这是 Amazon ECS 集群中实例的默认限制，该限制值可根据请求增大。

Note

目前，多节点并行作业中的所有节点组必须使用相同的实例类型。

# 作业生命周期

在提交多节点并行作业时，作业将进入 `SUBMITTED` 状态，并等待任何作业依赖项以完成操作。然后，作业将进入 `RUNNABLE` 状态，AWS Batch 将预配置所需的实例容量，以运行作业并启动这些实例。

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 将对其进行监控，以确定所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。

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，或者在应用程序代码内部协调等待，直到在开始任何分布式处理任务之前，所有节点均在线。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。如果某个子节点发生故障，AWS Batch 不会对作业中的其他节点采取任何操作。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，以终止或取消作业。

# 计算环境注意事项

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

- 如果打算将多节点并行作业提交到计算环境，请考虑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（适用于 Linux 实例）中的置放群组。
- 使用 Spot 实例的计算环境不支持多节点并行作业。
- AWS Batch 多节点并行作业采用 Amazon ECS `awsvpc` 网络模式，这为多节点并行作业容器提供了与 Amazon EC2 实例相同的联网属性。每个多节点并行作业容器都可获得自己的弹性网络接口、主要私有 IP 地址以及内部 DNS 主机名。在同一 VPC 子网中创建网络接口，作为其主机计算资源。适用于计算资

源的任何安全组，也适用于该主机计算资源。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 awsvpc 网络模式的任务联网。

- 与计算环境关联的安全组不得超过 5 个。
- 创建并附加到计算资源的弹性网络接口，不能由您的账户手动分离或修改。这是为了防止意外删除与正在运行的作业关联的弹性网络接口。要释放任务的弹性网络接口，请终止作业。
- 计算环境必须具有足够大的 vCPU 才能支持多节点并行作业。
- Amazon EC2 实例限制必须能够满足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，那么作业将会卡在 `RUNNABLE` 状态。
- 如果为多节点并行作业中的节点组指定了实例类型，那么计算环境必须能够启动该实例类型。

# GPU 作业

借助 GPU 作业，您可以运行使用实例 GPU 的作业。

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

| 实例类型 | GPU | GPU 内存 | vCPU | 内存 | 网络带宽 |
|---|---|---|---|---|---|
| g3s.xlarge | 1 | 8 GiB | 4 | 30.5 GiB | 10 Gbps |
| g3.4xlarge | 1 | 8 GiB | 16 | 122 GiB | 最高 10 Gbps |
| g3.8xlarge | 2 | 16 GiB | 32 | 244 GiB | 10 Gbps |
| g3.16xlarge | 4 | 32 GiB | 64 | 488 GiB | 25 Gbps |
| g4dn.xlarge | 1 | 16 GiB | 4 | 16 GiB | 最高 25 Gbps |
| g4dn.2xlarge | 1 | 16 GiB | 8 | 32 GiB | 最高 25 Gbps |
| g4dn.4xlarge | 1 | 16 GiB | 16 | 64 GiB | 最高 25 Gbps |
| g4dn.8xlarge | 1 | 16 GiB | 32 | 128 GiB | 50 Gbps |
| g4dn.12xlarge | 4 | 64 GiB | 48 | 192 GiB | 50 Gbps |
| g4dn.16xlarge | 1 | 16 GiB | 64 | 256 GiB | 50 Gbps |
| 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 |
| p3dn.24xlarge | 8 | 256 GiB | 96 | 768 GiB | 100 Gbps |

作业定义的resourceRequirements 参数指定要固定到容器并且在作业持续期间不可用于在实例上运行的任何其他作业的 GPU 的数量。计算环境中将运行 GPU 作业的所有实例类型都应来自 `p2`、`p3`、`g3`、`g3s` 或 `g4` 实例系列。如果不这么做，GPU 作业可能会停滞于 `RUNNABLE` 状态。

不使用 GPU 的作业可以在启用了 GPU 的实例上运行，但在启用了 GPU 的实例上运行的成本可能比在同样的非启用了 GPU 的实例上运行的成本要高。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。

# Job Definitions

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

内容：



在作业定义中指定的一些属性包括：

- Which Docker image to use with the container in your job
- How many vCPUs and how much memory to use with the container
- The command the container should run when it is started
- What (if any) environment variables should be passed to the container when it starts
- Any data volumes that should be used with the container
- What (if any) IAM role your job should use for AWS permissions

有关作业定义中可用的参数的完整描述，请参阅Job Definition Parameters (p. 36)。

# Creating a Job Definition

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。

要创建多节点并行作业定义，请参阅Creating a Multi-node Parallel Job Definition (p. 32)。有关多节点并行作业的更多信息，请参阅多节点并行作业 (p. 25)。

创建新的作业定义

1. 访问 https://console.aws.amazon.com/batch/，打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Job definitions 和 Create。
4. 对于 Job definition name，请为您的作业定义输入唯一名称。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. 对于 Job attempts，指定尝试作业的最大次数 (在尝试失败的情况下)。有关更多信息，请参阅 自动作业重试 (p. 16)。)
6. (可选) 对于执行超时，指定您希望允许任务尝试运行的最大秒数。如果尝试超过超时持续时间，则会停止并且状态移动到 `FAILED`。有关详细信息，请参阅 作业超时 (p. 16).
7. 对于 Job requires multiple node configurations (作业需要多个节点配置)，将此框保持未选中状态。要改为创建多节点并行作业定义，请参阅Creating a Multi-node Parallel Job Definition (p. 32)。

8. （可选）在 Parameters (参数) 部分中，您可以指定参数替代默认值和占位符，以便在您的作业容器启动时所运行的命令中使用。有关更多信息，请参阅 Parameters (p. 37)。）

   a. 选择 Add parameter (添加参数)。

   b. 对于 Key，指定参数的键。

   c. 对于 Value，指定参数的值。

9. （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 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-iam-roles.htmlAmazon Elastic Container Service Developer Guide 中的适用于任务的 IAM 角色。

   Note

   此处仅显示具有 Amazon Elastic Container Service 任务角色信任关系的角色。有关为 IAM 作业创建 AWS Batch 角色的更多信息，请参阅 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-iam-roles.html#create_task_iam_policy_and_roleAmazon Elastic Container Service Developer Guide 中的为任务创建 IAM 角色和策略。

10. 对于 Container image，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。您还可以指定其他存储库 *repository-url*/*image*:*tag*。 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。 此参数映射到 `Image` 在 创建容器 部分 Docker Remote API 和 `IMAGE` 参数 docker run.

    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。

    - Images in Amazon ECR repositories use the full `registry/repository:tag` naming convention. For example, *aws_account_id*`.dkr.ecr.`*region*`.amazonaws.com/`*my-web-app*`:`*latest*
    - Images in official repositories on Docker Hub use a single name (for example, `ubuntu` or `mongo`).
    - Images in other repositories on Docker Hub are qualified with an organization name (for example, `amazon/amazon-ecs-agent`).
    - Images in other online repositories are qualified further by a domain name (for example, `quay.io/assemblyline/ubuntu`).

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

    此参数映射到 `Cmd` 在 创建容器 部分 Docker Remote API 和 `COMMAND` 参数到 docker run。有关靠泊装置的更多信息 `CMD` 参数，转到 https://docs.docker.com/engine/reference/builder/#cmd.

    Note

    可以为您命令中的参数替代以及占位符使用默认值。有关更多信息，请参阅 Parameters (p. 37)。）

12. 对于 vCPUs，指定要为容器预留的 vCPU 数量。此参数映射到 `CpuShares` 在 创建容器 部分 Docker Remote API 和 `--cpu-shares` 选项 docker run。每个vCPU等同于1,024个CPU共享。您必须指定至少一个 vCPU。

13. 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。此参数映射到 `Memory` 在 创建容器 部分 Docker Remote API 和 `--memory` 选项 docker run。您必须为作业指定至少4MIB内存。

    Note

    如果您尝试通过为任务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，请参阅计算资源内存管理 (p. 71)。

14. （可选）在资源要求部分中，您可以配置作业容器的资源需求。（可选）对于 Number of GPU (GPU 数量)，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。

15. (可选) 在 Security 部分，可以为您的作业容器配置安全选项。

    a. 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 Privileged。此参数将映射到 Docker Remote API 的 创建容器 部分中的 `Privileged` 和 docker run 的 `--privileged` 选项。

    b. 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` 和 docker run 的 `--user` 选项。

16. （可选）在挂载点部分中，您可以配置作业容器要访问的挂载点。

    a. 对于 Container path，输入容器上挂载主机卷的路径。

    b. 对于 Source volume，输入要挂载的卷的名称。

    c. 要使该卷对容器呈现只读状态，请选择 Read-only。

17. （可选）在卷部分中，您可以为您的作业指定数据卷，以便传递到您的作业容器。

    a. 对于 Name (名称)，输入卷名称。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。

    b. (可选) 对于 Source Path，输入要提供给容器的主机实例的路径。如果您将此字段保留为空，则 Docker 守护程序将为您分配一个主机路径。如果您指定源路径，则数据卷将在主机容器实例上的指定位置保留，除非您手动将其删除。如果主机容器实例上不存在源路径，则 Docker 守护程序会创建它。如果该位置不存在，则源路径文件夹的内容将导出至容器。

18. （可选）在环境变量部分中，您可以指定要传递到您的作业容器的环境变量。此参数将映射到 Docker Remote API 的 创建容器 部分中的 `Env` 和 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。

    a. 选择 Add environment variable (添加环境变量)。

    b. 对于 Key，指定环境变量的键。

       Note

       环境变量不能以`AWS_BATCH` 开头；此命名约定是为 AWS Batch 服务所设置的变量保留的。

    c. 对于 Value，指定环境变量的值。

19. （可选）在 Ulimit 部分中，您可以配置要用于作业容器的任何 `ulimit` 值。

    a. 选择 Add limit (添加限制)。

    b. 对于 Limit name，选择要应用的 ulimit。

    c. 对于 Soft limit，选择要为 ulimit 类型应用的软限制。

    d. 对于 Hard limit，选择要为 ulimit 类型应用的硬限制。

20. （可选）在 Linux参数 部分中，您可以配置要用于作业容器的任何设备映射，以便容器可以访问主机实例上的设备。

    a. 在设备部分中，选择添加设备。

    b. 对于主机路径，指定主机实例中设备的路径。

    c. 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。

    d. 对于权限，选择要应用于容器中的设备的一个或多个权限。可用权限是 `READ`， `WRITE`，和 `MKNOD`.

21. 选择 Create job definition。

# Creating a Multi-node Parallel Job Definition

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。本主题介绍如何为 AWS Batch 多节点并行作业创建作业定义。有关更多信息，请参阅 多节点并行作业 (p. 25)。)

要创建单节点作业定义，请参阅Creating a Job Definition (p. 29)。

创建多节点并行作业定义

1. 访问 https://console.aws.amazon.com/batch/，打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Job definitions 和 Create。
4. 对于 Job definition name，请为您的作业定义输入唯一名称。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. 对于 Job attempts，指定尝试作业的最大次数 (在尝试失败的情况下)。有关更多信息，请参阅 自动作业重试 (p. 16)。)
6. (可选) 对于执行超时，指定您希望允许任务尝试运行的最大秒数。如果尝试超过超时持续时间，则会停止并且状态移动到 `FAILED`。有关详细信息，请参阅 作业超时 (p. 16).
7. 选择 Job requires multiple node configurations (作业需要多节点配置)，然后完成以下子步骤。要改为创建单节点并行作业定义，请参阅Creating a Job Definition (p. 29)。

   a. 对于 Number of nodes (节点数)，输入要用于作业的节点的总数。
   b. 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`。
   c. （可选）要将节点限制为特定实例类型，请从下拉菜单中选择一个。如果未指定实例类型，则 AWS Batch 从计算环境中的可用实例类型中选择满足最大节点（vCPU 和内存）要求的最小实例类型。

      Important

      请务必选择可在计算环境中启动的实例类型。否则，您的作业将卡在 `RUNNABLE` 状态并阻止后续作业。

8. （可选）在 Parameters (参数) 部分中，您可以指定参数替代默认值和占位符，以便在您的作业容器启动时所运行的命令中使用。有关更多信息，请参阅 Parameters (p. 37)。)

   a. 选择 Add parameter (添加参数)。
   b. 对于 Key，指定参数的键。
   c. 对于 Value，指定参数的值。

9. 在 Node properties (节点属性) 部分中，配置您的节点组。默认情况下，使用默认数量的节点为您创建单个节点组。
10. 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数，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。最终节点组的范围结束索引值应是您在Step 7.a (p. 32)中指定的节点数减 1。例如，如果指定了 10 个节点，并且要使用单个节点组，则结束范围应为 9。
11. 对于 Container image，选择要用于您的作业的 Docker 映像。默认情况下，Docker Hub 注册表中的映像可用。您还可以指定其他存储库 *`repository-url`*`/`*`image`*`:`*`tag`*。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。此参数映射到 `Image` 在 创建容器 部分 Docker Remote API 和 `IMAGE` 参数 docker run.

    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。

- Images in Amazon ECR repositories use the full `registry/repository:tag` naming convention. For example, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*
- Images in official repositories on Docker Hub use a single name (for example, `ubuntu` or `mongo`).
- Images in other repositories on Docker Hub are qualified with an organization name (for example, `amazon/amazon-ecs-agent`).
- Images in other online repositories are qualified further by a domain name (for example, `quay.io/assemblyline/ubuntu`).

12. 对于 vCPUs，指定要为容器预留的 vCPU 数量。此参数映射到 `CpuShares` 在 创建容器 部分 Docker Remote API 和 `--cpu-shares` 选项 docker run。每个vCPU等同于1,024个CPU共享。您必须指定至少一个 vCPU。
13. 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。此参数映射到 `Memory` 在 创建容器 部分 Docker Remote API 和 `--memory` 选项 docker run。您必须为作业指定至少4MIB内存。

    Note

    如果您尝试通过为任务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，请参阅计算资源内存管理 (p. 71)。
14. 对于 Command，指定要传递到容器的命令。对于简单的命令，您可以在 Space delimited 选项卡上键入命令，就像在命令提示符中键入命令一样。然后，确保 JSON 结果 (该结果将传递到 Docker 守护程序) 正确无误。对于较复杂的命令 (例如，带有特殊字符)，您可以切换到 JSON 选项卡，然后在该选项卡中输入等效字符串数组。

    此参数映射到 `Cmd` 在 创建容器 部分 Docker Remote API 和 `COMMAND` 参数到 docker run。有关靠泊装置的更多信息 `CMD` 参数，转到 https://docs.docker.com/engine/reference/builder/#cmd.

    Note

    可以为您命令中的参数替代以及占位符使用默认值。有关更多信息，请参阅 Parameters (p. 37)。)
15. （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 Privileged (特权)。此参数将映射到 Docker Remote API 的 创建容器 部分中的 `Privileged` 和 docker run 的 `--privileged` 选项。
16. （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 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-iam-roles.htmlAmazon Elastic Container Service Developer Guide 中的适用于任务的 IAM 角色。

    Note

    此处仅显示具有 Amazon Elastic Container Service 任务角色信任关系的角色。有关为 IAM 作业创建 AWS Batch 角色的更多信息，请参阅 https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-iam-roles.html#create_task_iam_policy_and_roleAmazon Elastic Container Service Developer Guide 中的为任务创建 IAM 角色和策略。
17. 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` 和 docker run 的 `--user` 选项。
18. (可选) 指定挂载点以供您的作业容器访问。
    a. 对于 Container path，输入容器上挂载主机卷的路径。
    b. 对于 Source volume，输入要挂载的卷的名称。
    c. 要使该卷对容器呈现只读状态，请选择 Read-only。
19. (可选) 可以为您的作业指定数据卷，以便传递到您的作业容器。
    a. 对于 Name (名称)，输入卷名称。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。

b. (可选) 对于 Source Path，输入要提供给容器的主机实例的路径。如果您将此字段保留为空，则 Docker 守护程序将为您分配一个主机路径。如果您指定源路径，则数据卷将在主机容器实例上的指定位置保留，除非您手动将其删除。如果主机容器实例上不存在源路径，则 Docker 守护程序会创建它。如果该位置不存在，则源路径文件夹的内容将导出至容器。

20. (可选) 您可以指定要传递到您的作业容器的环境变量。此参数将映射到 Docker Remote API 的 创建容器 部分中的 `Env` 和 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。

    a. 选择 Add environment variable (添加环境变量)。
    b. 对于 Key，指定环境变量的键。

       Note

       环境变量不能以`AWS_BATCH` 开头；此命名约定是为 AWS Batch 服务所设置的变量保留的。

    c. 对于 Value，指定环境变量的值。
21. 对于 Ulimits，配置用于您的作业容器的任何 ulimit 值。

    a. 选择 Add limit (添加限制)。
    b. 对于 Limit name，选择要应用的 ulimit。
    c. 对于 Soft limit，选择要为 ulimit 类型应用的软限制。
    d. 对于 Hard limit，选择要为 ulimit 类型应用的硬限制。
22. 返回Step 10 (p. 32)并对要为作业配置的每个节点组重复执行该步骤。
23. 选择 Create job definition。

# Job Definition Template

空作业定义模板如下所示。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 Job Definition Parameters (p. 36)。

```
{
    "jobDefinitionName": "",
    "type": "container",
    "parameters": {
        "KeyName": ""
    },
    "containerProperties": {
        "image": "",
        "vcpus": 0,
        "memory": 0,
        "command": [
            ""
        ],
        "jobRoleArn": "",
        "volumes": [
            {
                "host": {
                    "sourcePath": ""
                },
                "name": ""
            }
        ],
        "environment": [
```

```
            {
                "name": "",
                "value": ""
            }
        ],
        "mountPoints": [
            {
                "containerPath": "",
                "readOnly": true,
                "sourceVolume": ""
            }
        ],
        "readonlyRootFilesystem": true,
        "privileged": true,
        "ulimits": [
            {
                "hardLimit": 0,
                "name": "",
                "softLimit": 0
            }
        ],
        "user": "",
        "instanceType": "",
        "resourceRequirements": [
            {
                "value": "",
                "type": "GPU"
            }
        ],
        "linuxParameters": {
            "devices": [
                {
                    "hostPath": "",
                    "containerPath": "",
                    "permissions": [
                        "WRITE"
                    ]
                }
            ]
        }
    },
    "nodeProperties": {
        "numNodes": 0,
        "mainNode": 0,
        "nodeRangeProperties": [
            {
                "targetNodes": "",
                "container": {
                    "image": "",
                    "vcpus": 0,
                    "memory": 0,
                    "command": [
                        ""
                    ],
                    "jobRoleArn": "",
                    "volumes": [
                        {
                            "host": {
                                "sourcePath": ""
                            },
                            "name": ""
                        }
                    ],
                    "environment": [
                        {
                            "name": "",
```

```
                        "value": ""
                    }
                ],
                "mountPoints": [
                    {
                        "containerPath": "",
                        "readOnly": true,
                        "sourceVolume": ""
                    }
                ],
                "readonlyRootFilesystem": true,
                "privileged": true,
                "ulimits": [
                    {
                        "hardLimit": 0,
                        "name": "",
                        "softLimit": 0
                    }
                ],
                "user": "",
                "instanceType": "",
                "resourceRequirements": [
                    {
                        "value": "",
                        "type": "GPU"
                    }
                ],
                "linuxParameters": {
                    "devices": [
                        {
                            "hostPath": "",
                            "containerPath": "",
                            "permissions": [
                                "READ"
                            ]
                        }
                    ]
                }
            }
        ]
    },
    "retryStrategy": {
        "attempts": 0
    },
    "timeout": {
        "attemptDurationSeconds": 0
    }
}
```

Note

您可以使用以下 AWS CLI 命令生成上述作业定义模板：

```
$ aws batch register-job-definition --generate-cli-skeleton
```

# Job Definition Parameters

作业定义被拆分为四个基本部分：作业定义名称、作业定义的类型、参数替代占位符默认值以及作业的容器属性。

内容：



# Job Definition Name

`jobDefinitionName`

当您注册作业定义时，需要指定一个名称。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。使用该名称注册的第一个作业定义被指定为修订 1。注册到该名称下的任何后续作业定义将被指定增量修订号。

Type ：字符串

必填 是

# Type

`type`

当您注册作业定义时，需要指定作业类型。有关多节点并行作业（也称为阵列作业）的更多信息，请参阅the section called “Creating a Multi-node Parallel Job Definition” (p. 32)。

Type ：字符串

有效值：{、} `container` || `multinode`

必填 是

# Parameters

`parameters`

当您提交任务时，您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。这样，您就可以对多个作业使用相同的任务定义，这些作业在提交时使用相同的格式以编程方式在命令中更改值。

Type 字符串到字符串映射

必填 否

当您注册作业定义时，可以在作业容器属性的 `command` 字段中使用参数替代占位符。例如：。

```
"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`。使用作业定义中的 `parameters` 对象，您可以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`。

# Container Properties

当您注册作业定义时，必须指定容器属性列表，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 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Node Properties (p. 43)级别设置的。

`command`

传递给容器的命令。此参数映射到 `Cmd` 在 创建容器 部分 Docker Remote API 和 `COMMAND` 参数到 docker run。有关靠泊装置的更多信息 `CMD` 参数，参阅 https://docs.docker.com/engine/reference/builder/#cmd.

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

Type 字符串数组

必填 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` 和 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。

Type Key-ValuePairs阵列

必填 否

`name`

环境变量的名称。

Type ：字符串

必填 是，当 `environment` 使用。

`value`

环境变量的值。

Type ：字符串

必填 是，当 `environment` 使用。

```
"environment" : [
    { "name" : "string", "value" : "string" },
    { "name" : "string", "value" : "string" }
]
```

`image`

用于启动容器的映像。此字符串将直接传递给 Docker 守护程序。默认情况下，Docker Hub 注册表中的映像可用。您还可以指定其他存储库 *repository-url*/*image*:*tag*。 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。 此参数映射到 `Image` 在 创建容器 部分 Docker Remote API 和 `IMAGE` 参数 docker run.

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。

- Images in Amazon ECR repositories use the full `registry/repository:tag` naming convention. For example, *`aws_account_id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`my-web-app`*`:`*`latest`*
- Images in official repositories on Docker Hub use a single name (for example, `ubuntu` or `mongo`).
- Images in other repositories on Docker Hub are qualified with an organization name (for example, `amazon/amazon-ecs-agent`).
- Images in other online repositories are qualified further by a domain name (for example, `quay.io/assemblyline/ubuntu`).

Type ：字符串

必填 是

`jobRoleArn`

当您注册作业定义时，可以指定 IAM 角色。该角色向作业容器提供权限，以便代表您调用在关联的策略中指定的 API 操作。有关更多信息，请参阅 IAM 中的Amazon Elastic Container Service Developer Guide适用于任务的 角色。

Type ：字符串

必填 否

`memory`

要提供给容器的内存的硬限制（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。此参数映射到 `Memory` 在 创建容器 部分 Docker Remote API 和 `--memory` 选项 docker run。您必须为作业指定至少4MIB内存。这是必要的，但可以在多个节点并行(MNP)作业中指定；必须至少为每个节点指定一次。

Note

如果您尝试通过为任务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，请参阅计算资源内存管理 (p. 71)。

Type ：整数

必填 是

`mountPoints`

您的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` 和 docker run 的 `--volume` 选项。

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

Type 对象阵列

必填 否

`sourceVolume`

要挂载的卷的名称。

Type ：字符串

必填 是，当 `mountPoints` 使用。

`containerPath`

要将主机卷挂载到的容器上的路径。

Type ：字符串

必填 是，当 `mountPoints` 使用。

`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。默认值为 `false`。

Type 布尔型

必填 否

`privileged`

当该参数为 true 时，将对此容器提供对主机容器实例的提升的特权（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` 和 docker run 的 `--privileged` 选项。

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

Type 布尔型

必填 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` 和 docker run 的 `--read-only` 选项。

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

Type 布尔型

必填 否

`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` 和 docker run 的 `--ulimit` 选项。

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

Type 对象阵列

必填 否

name

TheThethe type 的 ulimit.

Type ：字符串

必填 是，当 ulimits 使用。

hardLimit

ulimit 类型的硬限制。

Type ：整数

必填 是，当 ulimits 使用。

softLimit

ulimit 类型的软限制。

Type ：整数

必填 是，当 ulimits 使用。

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 和 docker run 的 --user 选项。

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

Type ：字符串

必填 否

resourceRequirements

指示要为您的容器保留的 GPU 的数量。

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

Type 对象阵列

必填 否

type

GPU 是唯一受支持的值。

Type ：字符串

必填 是，当 resourceRequirements 使用。

value

每个容器将需要的物理 GPU 的数量。

Type ：字符串

必填 是，当 resourceRequirements 使用。

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"
                                ]
                            }
                        ]
                    }
```

Type Linux参数 对象

必填 否

devices

映射到容器的设备列表。

Type 阵列 设备 对象

必填 否

hostPath

主机中可用设备的路径。

Type ：字符串

必填 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。

Type ：字符串

必填 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。

Type ：字符串数组

必填 否

有效值：{、} READ || WRITE || MKNOD

vcpus

为容器预留的 vCPU 的数量。此参数映射到 CpuShares 在 创建容器 部分 Docker Remote API 和 --cpu-shares 选项 docker run。每个vCPU等同于1,024个CPU共享。必须至少指定一个vCPU。这是必要的，但可以在多个节点并行(MNP)作业中指定；必须至少为每个节点指定一次。

Type ：整数

必填 是

`volumes`

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

```
[
  {
    "name": "string",
    "host": {
      "sourcePath": "string"
    }
  }
]
```

`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。

Type ：字符串

必填 是

`host`

`host` 参数的内容确定您的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 守护程序将为您的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。

Type 对象

必填 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 守护程序将为您分配一个主机路径。

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，除非您手动将其删除。如果主机容器实例上不存在 `sourcePath` 值，则 Docker 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。

Type ：字符串

必填 否

# Node Properties

`nodeProperties`

在注册多节点并行作业定义时，您必须指定节点属性列表，这些属性定义要在作业中使用的节点数、主节点索引和要使用的各种节点范围。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 (p. 25)。)

Type Nodeproperties 对象

必填 否

`mainNode`

指定多节点并行作业的主节点的节点索引。

Type ：整数

必填 是

`numNodes`

与多节点并行作业关联的节点数。

Type ：整数

必填 是

`nodeRangeProperties`

与多节点并行作业关联的节点范围及其属性的列表。

Type 阵列 诺德兰地区 对象

必填 是

`targetNodes`

节点范围（使用节点索引值）。范围 0:3 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 来开始范围。如果省略结束范围值 (n:)，则使用可能最高的节点索引来结束范围。您的累积节点范围必须考虑所有节点 (0:n)。您可以嵌套节点范围，例如 0:10 和 4:5，在此情况下，4:5 范围属性会覆盖 0:10 属性。

Type ：字符串

必填 否

`container`

节点范围的容器详细信息。有关更多信息，请参阅 Container Properties (p. 38)。)

Type 容器属性 对象

必填 否

# Retry Strategy

`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，您可以选择性地指定要用于这些作业的重试策略。默认情况下，每个任务尝试一次。如果指定多次尝试，则在作业失败后 (例如，如果返回非零退出代码或者容器实例终止) 将会重试。有关更多信息，请参阅 自动作业重试 (p. 16)。)

Type 回缩策略 对象

必填 否

`attempts`

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

```
"attempts": integer
```

Type ：整数

必填 否

## Timeout

`timeout`

您可以为任务配置超时时间，以便在某个任务运行的时间超过超时时间时让 AWS Batch 终止该任务。有关更多信息，请参阅 作业超时 (p. 16)。）如果某个任务因超时而终止，它不会被重试。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 (p. 16)。）

Type 工作超时 对象

必填 否

`attemptDurationSeconds`

以秒为单位的持续时间（根据作业尝试的 `startedAt` 时间戳测得），在此时间过后，AWS Batch 将终止未完成的作业。超时时间的最小值为 60 秒。

Type ：整数

必填 否

# Example Job Definitions

以下示例作业定义说明了如何使用常见模式，如环境变量、参数替代和卷挂载。

## Use Environment Variables

以下示例作业定义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，但您可以提交使用此定义的作业，并指定命令和环境变量覆盖以使作业定义更加通用。

```
{
    "jobDefinitionName": "fetch_and_run",
    "type": "container",
    "containerProperties": {
        "image": "123456789012.dkr.ecr.us-east-1.amazonaws.com/fetch_and_run",
        "vcpus": 2,
        "memory": 2000,
        "command": [
            "myjob.sh",
            "60"
        ],
        "jobRoleArn": "arn:aws:iam::123456789012:role/AWSBatchS3ReadOnly",
        "environment": [
            {
                "name": "BATCH_FILE_S3_URL",
                "value": "s3://my-batch-scripts/myjob.sh"
            },
            {
                "name": "BATCH_FILE_TYPE",
                "value": "script"
            }
        ],
        "user": "nobody"
    }
```

```
}
```

# Using Parameter Substitution

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::` 部分中的 `command` 声明用于设置参数替代的占位符。当您在此工作定义中提交一个工作时，您会指定参数覆盖以填写这些值，例如 `inputfile` 和 `outputfile`。TheThethe `parameters` 以下部分设置默认值 `codec`，但您可以根据需要覆盖该参数。

有关更多信息，请参阅 Parameters (p. 37)。)

```
{
    "jobDefinitionName": "ffmpeg_parameters",
    "type": "container",
    "parameters": {"codec": "mp4"},
    "containerProperties": {
        "image": "my_repo/ffmpeg",
        "vcpus": 2,
        "memory": 2000,
        "command": [
            "ffmpeg",
            "-i",
            "Ref::inputfile",
            "-c",
            "Ref::codec",
            "-o",
            "Ref::outputfile"
        ],
        "jobRoleArn": "arn:aws:iam::123456789012:role/ECSTask-S3FullAccess",
        "user": "nobody"
    }
}
```

# Test GPU Functionality

在以下示例中，作业定义测试使用 GPU 工作负载 AMI (p. 56) 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",
        "vcpus": 8,
        "command": [
            "sh",
            "-c",
            "cd /tensorflow/tensorflow/examples/tutorials/mnist; python mnist_deep.py"
        ],
        "memory": 32000
    },
    "type": "container",
    "jobDefinitionName": "tensorflow_mnist_deep"
}
```

您可以创建名为 `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 任务定义：

```
aws batch register-job-definition --cli-input-json file://tensorflow_mnist_deep.json
```

# Multi-node Parallel Job

以下作业定义示例描述了多节点并行作业。有关详细信息，请参阅 在多节点平行工作中构建紧密耦合的 MolecularDynamics工作流 AWS Batch 在 AWS Compute 博客。

```
{
  "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",
          "vcpus": 8,
          "memory": 24000,
          "command": [],
          "jobRoleArn": "arn:aws:iam::123456789012:role/ecsTaskExecutionRole",
          "ulimits": [],
          "instanceType": "p3.2xlarge"
        }
      }
    ]
  }
}
```

# 作业队列

作业将提交到作业队列，它们将一直位于队列中，直到能够在计算环境中计划运行这些作业。一个 AWS 账户可以有多个作业队列。例如，您可创建一个将 Amazon EC2 按需实例用于高优先级任务的队列；并创建另一个将 Amazon EC2 Spot 实例用于低优先级任务的队列。作业队列具有优先级，计划程序使用该优先级来确定应评估哪个队列中的作业首先执行。

## 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，并且为计算环境分配优先顺序。

您还可以设置作业队列优先级，该优先级决定了 AWS Batch 计划程序在关联的计算环境中置放作业的顺序。例如，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在该计算环境中计划作业。

创建作业队列

1. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Job queues 和 Create queue。
4. 对于 Queue name，为作业队列输入唯一名称。
5. 请务必选择 Enable job queue，以便您的作业队列能够接受作业提交。
6. 对于 Priority，为作业队列的优先级输入一个整数值。当有多个作业队列与同一计算环境关联时，系统将首先评估具有较高优先级 (或 `priority` 参数的较高整数值) 的作业队列。优先级按降序顺序确定，例如，优先级值为 10 的作业队列将会比优先级值为 1 的作业队列优先计划。
7. 在 Connected compute environments for this queue 部分，从列表中按照队列置放的尝试顺序选择一个或多个计算环境，以便与作业队列关联。作业计划程序使用计算环境顺序来确定哪个计算环境应执行给定作业。计算环境必须先处于 `VALID` 状态，然后您才能将其与作业队列关联。您最多可以将三个计算环境与一个作业队列关联。

   Note

   与作业队列关联的所有计算环境必须共享相同架构。AWS Batch 不支持在单个作业队列中混用计算环境架构类型。

   您可以通过选择表中 Order 列旁边的向上和向下箭头，来更改计算环境的顺序。
8. 选择 Create 以完成和创建作业队列。

## 作业队列模板

空作业队列模板如下所示。您可以使用此模板创建任务队列，随后可将任务队列保存到文件并与 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 CreateJobQueue。

```
{
```

```
    "jobQueueName": "",
    "state": "",
    "priority": 0,
    "computeEnvironmentOrder": [{
        "order": 0,
        "computeEnvironment": ""
    }]
}
```

Note

您可以使用以下 AWS CLI 命令生成上述作业队列模板。

```
$ aws batch create-job-queue --generate-cli-skeleton
```

# 作业队列参数

作业队列被拆分为四个基本组成部分：作业队列的名称、状态和优先级，以及计算环境顺序。

## 作业队列名称

`jobQueueName`

作业队列的名称。允许使用不超过 128 个字母 (大写和小写字母)、数字和下划线。

类型：字符串

必需：是

## State

`state`

作业队列的状态。如果作业队列状态为 `ENABLED` (默认值)，它可以接受作业。

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

## 优先级

`priority`

作业队列的优先级。当有多个作业队列与同一计算环境关联时，系统将首先评估具有较高优先级 (或 `priority` 参数的较高整数值) 的作业队列。优先级按降序顺序确定，例如，优先级值为 10 的作业队列将会比优先级值为 1 的作业队列优先计划。

类型：整数

必需：是

# 计算环境顺序

`computeEnvironmentOrder`

一组计算环境，它们映射到一个作业队列，并且其顺序是彼此相关的。作业计划程序使用此参数来确定哪个计算环境应执行给定作业。计算环境必须先处于 `VALID` 状态，然后您才能将其与作业队列关联。您最多可以将三个计算环境与一个作业队列关联。

> **Note**
>
> 与作业队列关联的所有计算环境必须共享相同架构。AWS Batch 不支持在单个作业队列中混用计算环境架构类型。

类型：ComputeEnvironmentOrder 对象的数组

必需：是

`computeEnvironment`

计算环境的 Amazon 资源名称 (ARN)。

类型：字符串

必需：是

`order`

计算环境的顺序。计算环境按照升序顺序尝试。例如，如果有两个计算环境与一个作业队列关联，则具有较低 `order` 整数值的计算环境首先尝试进行作业置放。

# 作业计划

AWS Batch 计划程序评估何时、何地以及如何运行已提交到作业队列的作业。只要满足其他作业的所有依赖关系，就会按作业的大致提交顺序运行作业。

# 计算环境

作业队列映射到一个或多个计算环境。计算环境包含用于运行容器化批处理作业的 Amazon ECS 容器实例。还可以将一个给定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，计划程序使用此顺序来确定用于置放准备好执行的作业的位置。如果第一个计算环境具有免费的资源，则针对该计算环境中的一个容器实例计划作业。如果计算环境无法提供合适的计算资源，计划程序将尝试在下一个计算环境中运行作业。

主题



# 托管计算环境

利用托管计算环境，您可以描述您的业务要求。在托管计算环境中，AWS Batch 将根据您在创建计算环境时定义的计算资源规范来管理环境中的计算资源的容量和实例类型。您可以选择在您的托管计算环境中使用 Amazon EC2 按需实例或 Spot 实例。您可以选择设置最高价，以便 Spot 实例仅在 Spot 实例价格低于按需价格的某个指定百分比时启动。

托管计算环境在您创建计算环境时指定的 VPC 和子网中启动 Amazon ECS 容器实例。Amazon ECS 容器实例需要外部网络访问以便与 Amazon ECS 服务终端节点进行通信。如果您的容器实例没有公有 IP 地址 (因为您选择的子网在默认情况下不提供该地址)，则容器实例必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关。要获取有关创建 VPC 的帮助，请参阅教程：为您的计算环境创建带有公有和私有子网的 VPC (p. 100)。

默认情况下，AWS Batch 托管计算环境将最近批准的经 Amazon ECS 优化的 AMI 版本用于计算资源。但是，您可能出于各种原因需要创建自己的 AMI 以用于您的托管计算环境。有关更多信息，请参阅计算资源 AMI (p. 53)。

> **Note**
>
> AWS Batch 不会升级计算环境中的已创建 AMI（例如，当较新版本的经 Amazon ECS 优化的 AMI 可用时）。您负责管理来宾操作系统（包括更新和安全补丁）以及您在计算资源上安装的任何其他应用程序软件或实用程序。要为您的 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. 从您的作业队列中删除旧的计算环境。
> 4. 删除旧的计算环境。

# 非托管计算环境

在非托管计算环境中，您需要管理自己的计算资源。您必须确保用于计算资源的 AMI 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 (p. 53) 和 创建计算资源 AMI (p. 54)。

在创建非托管计算环境后，可使用 DescribeComputeEnvironments API 操作查看计算环境详细信息。找到与环境关联的 Amazon ECS 集群，然后手动在该 Amazon ECS 集群中启动您的容器实例。

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 Developer Guide 中的启动 Amazon ECS 容器实例。当您启动计算资源时，使用以下 Amazon EC2 用户数据指定资源将注册到的 Amazon ECS 集群 ARN。将 *ecsClusterArn* 替换为使用上一条命令获取的集群 ARN。

```
#!/bin/bash
echo "ECS_CLUSTER=ecsClusterArn" >> /etc/ecs/ecs.config
```

# 计算资源 AMI

默认情况下，AWS Batch 托管计算环境将最近批准的经 Amazon ECS 优化的 AMI 版本用于计算资源。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：

- 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
- 使用自定义选项配置 Amazon ECS 容器代理
- 将 Docker 配置为使用自定义选项
- 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

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。

  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（而不是通过指定自定义 AMI）。有关更多信息，请参阅多节点并行作业 (p. 25)。
- Amazon ECS 容器代理（最好是最新版本）。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 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 Developer Guide 中的 Amazon ECS 容器代理配置。
- 运行至少 1.9 版的 Docker 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 代理版本附带的且经测试的 Docker 版本。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 容器代理版本。

推荐使用

- 用于运行和监控 Amazon ECS 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`。要查看使用 `systemd` 启动和监视 Amazon ECS 容器代理的几个示例用户数据配置脚本，请参阅 Amazon Elastic Container Service Developer Guide 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 Developer Guide 中的经 Amazon ECS 优化的 AMI。

# 创建计算资源 AMI

您可以创建您自己的自定义计算资源 AMI 以用于托管计算环境和非托管计算环境，前提是您遵循计算资源 AMI 规范 (p. 53)。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。

创建自定义计算资源 AMI

1. 选择从中启动的基本 AMI。基本 AMI 必须使用 HVM 虚拟化，并且不能是 Windows AMI。

   Note

   为一个计算环境选择的 AMI 必须与要为该计算环境使用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，您选择的计算资源 AMI 必须支持 ARM 实例。Amazon ECS 同时提供经 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 ARM 版本。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的经 Amazon ECS 优化的 Amazon Linux 2 AMI。

   经 Amazon ECS 优化的 AMI 是托管计算环境中的计算资源的默认 AMI。AWS 工程师将在 AWS Batch 上预配置和测试经 Amazon ECS 优化的 AMI。这是可供您开始操作并快速获取 AWS 上运行的计算资源的最简单 AMI。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的经 Amazon ECS 优化的 AMI。

   或者，您可以选择另一个 Amazon Linux 变体，并使用以下命令安装 `ecs-init` 程序包：

   ```
   sudo yum install -y ecs-init
   ```

   例如，如果您需要在 AWS Batch 计算资源上运行 GPU 工作负载，可以首先使用 Amazon Linux Deep Learning AMI，并对其进行配置以便能够运行AWS Batch 任务。有关更多信息，请参阅使用 GPU 工作负载 AMI (p. 56)。

Important

如果您选择不支持 `ecs-init` 程序包的基本 AMI，则必须配置一种方式以使 Amazon ECS 代理在系统启动时启动并确保其保持运行状态。要查看使用 `systemd` 启动和监视 Amazon ECS 容器代理的几个示例用户数据配置脚本，请参阅 Amazon Elastic Container Service Developer Guide 中的示例容器实例用户数据配置脚本。

2. 使用适用于 AMI 的存储选项从选定的基本 AMI 启动实例。您可以配置附加的 Amazon EBS 卷或实例存储卷 (如果选定实例类型支持实例存储卷) 的大小和数量。有关更多信息，请参阅 Amazon EC2 用户指南 ( 适用于 Linux 实例 ) 中的启动实例和 Amazon EC2 实例存储。
3. 使用 SSH 连接到您的实例并执行任何必要的配置任务，例如：
   - 安装 Amazon ECS 容器代理。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 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 ( 适用于 Linux 实例 ) 中的使用 SSH 连接到您的 Linux 实例。
4. 如果您在实例上启动了 Amazon ECS 容器代理，则必须先停止它并在创建 AMI 之前删除持久数据检查点文件；否则，代理将不会在从您的 AMI 启动的实例上启动。
   a. 停止 Amazon ECS 容器代理。
      - Amazon ECS-optimized Amazon Linux 2 AMI：

      ```
      sudo systemctl stop ecs
      ```

      - Amazon ECS-optimized Amazon Linux AMI：

      ```
      sudo stop ecs
      ```

   b. 删除持久性数据检查点文件。默认情况下，此文件位于 `/var/lib/ecs/data/ecs_agent_data.json` 中。可使用以下命令删除此文件。

   ```
   sudo rm -rf /var/lib/ecs/data/ecs_agent_data.json
   ```

5. 从正在运行的实例创建新的 AMI。有关更多信息，请参阅 Amazon EC2 用户指南 ( 适用于 Linux 实例 ) 指南中的创建 Amazon EBS 支持的 Linux AMI。

### 将新 AMI 与 AWS Batch 结合使用

1. 当 AMI 创建过程完成时，请使用新 AMI 创建计算环境 AMI ( 确保选择启用用户指定的 AMI ID 并在Step 5.j (p. 64) 中指定自定义 AMI ID )。有关更多信息，请参阅创建计算环境 (p. 63)。

   Note

   为一个计算环境选择的 AMI 必须与要为该计算环境使用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，您选择的计算资源 AMI 必须支持 ARM 实例。Amazon ECS 同时提供经 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 ARM 版本。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的经 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 (p. 48)。

Note

与作业队列关联的所有计算环境必须共享相同架构。AWS Batch 不支持在单个作业队列中混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Example Job Definitions (p. 45)、Creating a Job Definition (p. 29)和提交作业 (p. 12)。

# 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 Developer Guide 中的在 Amazon ECS 上使用 GPU 和经过 Amazon ECS 优化的 AMI。

在托管计算环境中，如果计算环境指定任何 `p2`、`p3`、`g3`、`g3s` 或 `g4` 实例类型或实例系列，则 AWS Batch 使用经过 Amazon ECS GPU 优化的 AMI。

在非托管计算环境中，建议使用经过 Amazon ECS GPU 优化的 AMI。可以使用 AWS Command Line Interface 或 AWS Systems Manager Parameter Store GetParameter、GetParameters 和 GetParametersByPath 操作来检索元数据，从而获得建议的经过 Amazon ECS GPU 优化的 AMI。

以下示例演示 GetParameter 的用法。

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，使用完整路径名称：

```
/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
```

以下示例演示 GetParametersByPath 的用法。

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 Developer Guide 中的检索经过 Amazon ECS 优化的 AMI 元数据。

# 启动模板支持

AWS Batch 支持在计算环境中使用 Amazon EC2 启动模板。启动模板支持允许您修改 AWS Batch 计算资源的默认配置，而无需您创建自定义的 AMI。

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 控制台中创建启动模板，也可以使用 AWS CLI 或 AWS 开发工具包。例如，下面的 JSON 文件表示一个启动模板，它为默认 AWS Batch 计算资源 AMI 调整 Docker 数据卷的大小，并将其设置为加密。

```
{
    "LaunchTemplateName": "increase-container-volume-encrypt",
    "LaunchTemplateData": {
        "BlockDeviceMappings": [
            {
                "DeviceName": "/dev/xvdcz",
                "Ebs": {
                    "Encrypted": true,
                    "VolumeSize": 100,
                    "VolumeType": "gp2"
                }
            }
        ]
    }
}
```

您可以通过将 JSON 保存到名为 `lt-data.json` 的文件并运行以下 AWS CLI 命令来创建上述启动模板：

```
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（适用于 Linux 实例）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**
>
> 如果在启动模板和计算环境配置中都指定了这些参数中的任何一个（Amazon EC2 标签除外），则计算环境参数优先。Amazon EC2 标签将在启动模板和计算环境配置之间合并。如果标签键发生冲突，则计算环境配置中的值优先。

- Amazon EC2 密钥对
- Amazon EC2 AMI ID
- 安全组 ID
- Amazon EC2 标签

AWS Batch 将忽略以下启动模板参数：

- 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 Spot 实例配置）
- 禁用 API 终止（AWS Batch 必须控制实例生命周期）

AWS Batch 不支持使用新的启动模板版本来更新计算环境。如果更新启动模板，则必须使用新模板创建新计算环境以使更改生效。

# 启动模板中的 Amazon EC2 用户数据

可以在启动模板中提供当实例启动时由 cloud-init 执行的 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

启动模板中的 Amazon EC2 用户数据必须采用 MIME 分段存档格式，因为您的用户数据将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 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。

如果您使用的是 AWS CloudFormation，AWS::CloudFormation::Init 类型可以与 cfn-init 帮助程序脚本一起使用，以执行常见的配置方案。

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==`
  - 数据块的内容类型声明：`Content-Type: text/cloud-config; charset="us-ascii"`。有关内容类型的更多信息，请参阅 Cloud-Init 文档。
  - 用户数据的内容，例如，Shell 命令或 `cloud-init` 指令的列表
- 封闭边界，表示 MIME 分段文件的结尾：`--==BOUNDARY==--`

以下是一些示例 MIME 分段文件，您可以使用它们来创建您自己的文件。

> **Note**
>
> 如果将用户数据添加到 Amazon EC2 控制台中的启动模板，则可以将其作为纯文本粘贴或从文件进行上载。如果您使用 AWS CLI 或 AWS 开发工具包，则必须先对用户数据进行 `base64` 编码，并在调用 CreateLaunchTemplate 时将该字符串作为 `UserData` 参数值提交，如下面的 JSON 中所示。

```
{
    "LaunchTemplateName": "base64-user-data",
    "LaunchTemplateData": {
        "UserData":
 "ewogICAgIkxhdW5jaFRlbXBsYXRlTmFtZSI6ICJpbmNyZWFzZS1jb250YWluZXItdm9sdW..."
    }
}
```

示例

- 示例：挂载现有 Amazon EFS 文件系统 (p. 61)
- 示例：覆盖默认 Amazon ECS 容器代理配置 (p. 62)
- 示例：挂载现有 Amazon FSx for Lustre 文件系统 (p. 62)

# 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 Amazon FSx for Lustre 文件系统

Example

此示例 MIME 分段文件将配置计算资源，以从 Extras 库安装 `lustre2.10` 程序包，并在 `/scratch` 处装载现有 Amazon FSx for Lustre 文件系统。此示例适用于 Amazon Linux 2。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 。

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--===MYBOUNDARY==
Content-Type: text/cloud-config; charset="us-ascii"

runcmd:
- file_system_id_01=fs-0abcdef1234567890
- region=us-east-2
- fsx_directory=/scratch
- amazon-linux-extras install -y lustre2.10
- mkdir -p ${fsx_directory}
- mount -t lustre ${file_system_id_01}.fsx.${region}.amazonaws.com@tcp:fsx ${fsx_directory}

--===MYBOUNDARY==--
```

在容器属性的 volumes 和 mountPoints 成员中，装载点必须映射到容器中。

```
{
    "volumes": [
        {
            "host": {
                "sourcePath": "/scratch"
            },
```

```
            "name": "Scratch"
        }
    ],
    "mountPoints": [
        {
            "containerPath": "/scratch",
            "sourceVolume": "Scratch"
        }
    ],
}
```

# 创建计算环境

您需要先创建计算环境，然后才能在 AWS Batch 中运行作业。您可以创建一个托管计算环境，其中 AWS Batch 会基于您的规范管理环境中的实例；您也可以创建一个非托管计算环境，您可在该环境中处理实例配置。

### 创建托管计算环境

1. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Compute environments 和 Create environment。
4. 配置环境。
    a. 对于 Compute environment type，选择 Managed。
    b. 对于 Compute environment name，为您的计算环境指定唯一名称。您可以使用最多 128 个字母 (大写形式和小写形式)、数字、连字符和下划线。
    c. 对于 Service role，选择创建新角色或使用现有角色。该角色允许 AWS Batch 服务代表您调用所需的 AWS API。有关更多信息，请参阅AWS Batch Service IAM Role (p. 83)。如果您选择创建新角色，则将为您创建所需的角色 (`AWSBatchServiceRole`)。
    d. 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。有关更多信息，请参阅Amazon ECS Instance Role (p. 86)。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`)。
    e. 对于 EC2 密钥对，选择在启动时与该实例关联的现有 Amazon EC2 密钥对。利用此密钥对，您可以使用 SSH 连接到实例 (确保安全组允许通过端口 22 访问)。
    f. 请务必选择启用计算环境，使计算环境能够接受来自 AWS Batch 任务计划程序的任务。
5. 配置您的计算资源。
    a. 对于预配置模型，选择按需以启动 Amazon EC2 按需实例，或选择 Spot 以使用 Amazon EC2 Spot 实例。
    b. 在选择使用 Spot 实例的情况下：
        i. (可选) 对于最高价，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 Spot 实例价格可达到的最大百分比。例如，如果最高价为 20%，则 Spot 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。
        ii. 对于 Spot 队组角色，选择一个现有 Amazon EC2 Spot 队组 IAM 角色以应用于您的 Spot 计算环境。如果您没有现有的 Amazon EC2 Spot 队组 IAM 角色，则必须先创建一个。有关更多信息，请参阅Amazon EC2 Spot Fleet Role (p. 86)。

            Important

            要在创建时标记 Spot 实例 ( 请参阅Step 7 (p. 65) )，您的 Amazon EC2 Spot 队组 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 Spot 实例所需的权限。有关更多信息，请参阅在创建时未标记的 Spot 实例 (p. 115)。

c. 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启动这些系列中的任何实例类型（例如，`c5`、`c5n` 或 `p3`），也可以指定系列中的特定大小（例如，`c5.8xlarge`）。请注意，裸机实例类型不包括在实例系列中（例如，`c5` 不包括 `c5.metal`）。 您也可以选择 `optimal` 以（从最新的 C、M 和 R 实例系列中）动态选取符合作业队列要求的实例类型。

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 计划，计算环境必须包含 `p2`、`p3`、`g3`、`g3s` 或 `g4` 系列的实例类型。

d. 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。有关更多信息，请参阅the section called “分配策略” (p. 71)。

e. （可选）对于 Launch template (启动模板)，选择现有的 Amazon EC2 启动模板以配置计算资源；将自动填充默认版本的模板。有关更多信息，请参阅启动模板支持 (p. 59)。

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

Important

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

g. 对于 Minimum vCPUs，选择您的计算环境应保留的 EC2 vCPU 的最小数目，而无论作业队列需求如何。

h. 对于 Desired vCPUs，请选择您的计算环境在启动时应使用的 EC2 vCPU 数量。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)。

i. 对于 Maximum vCPUs，选择您的计算环境可以向外扩展到的 EC2 vCPU 的最大数目，而无论作业队列需求如何。

j. （可选）选中启用用户指定的 AMI ID 以使用您自己的自定义 AMI。默认情况下，AWS Batch 托管计算环境将最近批准的经 Amazon ECS 优化的 AMI 版本用于计算资源。您可以根据计算资源 AMI 规范，在计算环境中创建和使用您自己的 AMI。有关更多信息，请参阅计算资源 AMI (p. 53)。

Note

为一个计算环境选择的 AMI 必须与要为该计算环境使用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，您选择的计算资源 AMI 必须支持 ARM 实例。Amazon ECS 同时提供经 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 ARM 版本。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的经 Amazon ECS 优化的 Amazon Linux 2 AMI。

- 对于 AMI ID，粘贴您的自定义 AMI ID，然后选择验证 AMI。

6. 配置网络。

Important

计算资源需要访问权限以便与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 终端节点或通过具有公有 IP 地址的计算资源完成。
有关接口 VPC 终端节点的更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink)。

如果您没有配置接口 VPC 终端节点并且您的计算资源没有公有 IP 地址，则它们必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关。有关更多信息，请参阅教程：为您的计算环境创建带有公有和私有子网的 VPC (p. 100)。

a. 对于 VPC ID，选择在其中启动实例的 VPC。
b. 对于 Subnets，选择选定 VPC 中应托管实例的子网。默认情况下，将选择选定 VPC 中的所有子网。
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。

7. (可选) 为实例添加标签。例如，您可以指定 `"Name": "AWS Batch Instance - C4OnDemand"` 作为标签，以便计算环境中的每个实例均具有此名称。这对于在 Amazon EC2 控制台中识别您的 AWS Batch 实例很有用。
8. 选择 Create 以完成。

### 创建非托管计算环境

1. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Compute environments 和 Create environment。
4. 对于 Compute environment type，选择 Unmanaged。
5. 对于 Compute environment name，为您的计算环境指定唯一名称。您可以使用最多 128 个字母 (大写形式和小写形式)、数字、连字符和下划线。
6. 对于服务角色，选择创建新角色或使用现有角色，后者允许 AWS Batch 服务代表您调用所需的 AWS API。有关更多信息，请参阅AWS Batch Service IAM Role (p. 83)。如果您选择创建新角色，则将为您创建所需的角色 (`AWSBatchServiceRole`)。
7. 请务必选择启用计算环境，使计算环境能够接受来自 AWS Batch 任务计划程序的任务。
8. 选择 Create 以完成。
9. (可选) 在 Amazon ECS 集群 ARN 中检索关联集群。以下 AWS CLI 命令提供计算环境的 Amazon ECS 集群 ARN：

```
aws batch describe-compute-environments --compute-environments unmanagedCE --query
 computeEnvironments[].ecsClusterArn
```

10. (可选) 在关联的 Amazon ECS 集群中启动容器实例。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的启动 Amazon ECS 容器实例。当您启动计算资源时，使用以下 Amazon EC2 用户数据指定资源将注册到的 Amazon ECS 集群 ARN。将 *ecsClusterArn* 替换为使用上一条命令获取的集群 ARN。

```
#!/bin/bash
echo "ECS_CLUSTER=ecsClusterArn" >> /etc/ecs/ecs.config
```

Note

您的非托管计算环境不具有任何计算资源，直到您手动启动它们。

# 计算环境模板

空计算环境模板如下所示。您可以使用此模板创建计算环境，随后可将计算环境保存到文件并与 AWS CLI `--cli-input-json` 选项结合使用。有关这些参数的更多信息，请参阅 AWS Batch API 参考 中的 CreateComputeEnvironment。

```
{
    "computeEnvironmentName": "",
    "type": "UNMANAGED",
    "state": "ENABLED",
    "computeResources": {
        "type": "SPOT",
        "allocationStrategy": "SPOT_CAPACITY_OPTIMIZED",
        "minvCpus": 0,
        "maxvCpus": 0,
        "desiredvCpus": 0,
        "instanceTypes": [
            ""
        ],
        "imageId": "",
        "subnets": [
            ""
        ],
        "securityGroupIds": [
            ""
        ],
        "ec2KeyPair": "",
        "instanceRole": "",
        "tags": {
            "KeyName": ""
        },
        "placementGroup": "",
        "bidPercentage": 0,
        "spotIamFleetRole": "",
        "launchTemplate": {
            "launchTemplateId": "",
            "launchTemplateName": "",
            "version": ""
        }
    },
    "serviceRole": ""
}
```

Note

您可以使用以下 AWS CLI 命令生成上述计算环境模板。

```
$ aws batch create-compute-environment --generate-cli-skeleton
```

# 计算环境参数

计算环境拆分为 5 个基本组成部分：计算环境的名称、类型和状态，计算资源定义（如果计算环境为托管计算环境）以及服务角色（用于向 AWS Batch 提供 IAM 权限）。

主题

# 计算环境名称

`computeEnvironmentName`

您的计算环境的名称。您可以使用最多 128 个字母 (大写形式和小写形式)、数字、连字符和下划线。

类型：字符串

必需：是

# 类型

`type`

计算环境的类型。选择 `MANAGED` 可让 AWS Batch 管理您定义的计算资源。有关更多信息，请参阅计算资源 (p. 67)。选择 `UNMANAGED` 可管理您自己的计算资源。

类型：字符串

有效值：`MANAGED` | `UNMANAGED`

必需：是

# State

`state`

计算环境的状态。

如果状态为 `ENABLED`，则 AWS Batch 计划程序可以尝试将关联任务队列中的任务放到环境中的计算资源上。如果计算环境是托管计算环境，则此环境可根据作业队列需求，自动向外扩展或缩减其实例。

如果状态为 `DISABLED`，则 AWS Batch 计划程序不会尝试将任务放在环境中。处于 `STARTING` 或 `RUNNING` 状态的作业将继续正常运行。处于 `DISABLED` 状态的托管计算环境不会向外扩展；但是，一旦实例变为空闲状态，这些环境将缩减到最低实例数以满足 `minvCpus` 值。

类型：字符串

有效值：`ENABLED` | `DISABLED`

必需：否

# 计算资源

`computeResources`

由计算环境托管的计算资源的详细信息。

类型：ComputeResource 对象

必需：此参数是托管计算环境所必需的

`type`

计算环境的类型。使用此参数可指定在计算环境中使用 Amazon EC2 按需实例还是 Amazon EC2 Spot 实例。如果您选择 `SPOT`，则还必须使用 `spotIamFleetRole` 参数指定 Amazon EC2 Spot 队组角色。有关更多信息，请参阅Amazon EC2 Spot Fleet Role (p. 86)。

有效值：EC2 | SPOT

必需：是

allocationStrategy

该分配策略在无法分配最佳匹配实例类型的足够实例时，用于计算资源。这可能是由于实例类型在该区域的可用性或 Amazon EC2 服务限制所致。如果未指定，则默认值为 BEST_FIT，该值将仅使用最佳匹配实例类型；如果没有最佳匹配实例类型可用，则等待额外的容量。此分配策略可降低成本，但会限制扩展。BEST_FIT_PROGRESSIVE 将选择足够大的、能够满足队列中作业要求的额外实例类型，并优先选择成本更低的实例类型。SPOT_CAPACITY_OPTIMIZED 仅适用于 Spot 实例计算资源，并将选择足够大的、能够满足队列中作业要求的额外实例类型，并优先选择不太可能会中断的实例类型。

有效值：BEST_FIT | BEST_FIT_PROGRESSIVE | SPOT_CAPACITY_OPTIMIZED

必需：否

minvCpus

环境应保留的 Amazon EC2 vCPU 的最小数量（即使计算环境处于 DISABLED 状态）。

类型：整数

必需：是

maxvCpus

环境可达到的最大 Amazon EC2 vCPU 数。

Note

同时使用 BEST_FIT_PROGRESSIVE 和 SPOT_CAPACITY_OPTIMIZED 分配策略时，AWS Batch 可能需要超过 maxvCpus 以满足您的容量要求。在这种情况下，AWS Batch 永远不会超过 maxvCpus 一个实例（例如，在计算环境中指定的实例中，不超过一个实例）。

类型：整数

必需：是

desiredvCpus

计算环境中所需的 Amazon EC2 vCPU 数。AWS Batch 根据任务队列需求，将此值修改为介于最小值和最大值之间。

类型：整数

必需：否

instanceTypes

可启动的实例类型。您可以指定实例系列以启动这些系列中的任何实例类型（例如，c5、c5n 或 p3），也可以指定系列中的特定大小（例如，c5.8xlarge）。请注意，裸机实例类型不包括在实例系列中（例如，c5 不包括 c5.metal）。 您也可以选择 optimal 以（从最新的 C、M 和 R 实例系列中）动态选取符合作业队列要求的实例类型。

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。

类型：字符串数组

必需：是

`imageId`

用于计算环境中启动的实例的 Amazon 系统映像 (AMI) ID。

Note

为一个计算环境选择的 AMI 必须与要为该计算环境使用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，您选择的计算资源 AMI 必须支持 ARM 实例。Amazon ECS 同时提供经 Amazon ECS 优化的 Amazon Linux 2 AMI 的 x86 和 ARM 版本。有关更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的经 Amazon ECS 优化的 Amazon Linux 2 AMI。

类型：字符串

必需：否

`subnets`

计算资源在其中启动的 VPC 子网。这些子网必须位于同一 VPC 中。

类型：字符串数组

必需：是

`securityGroupIds`

要与计算环境中启动的实例关联的 EC2 安全组。

类型：字符串数组

必需：是

`ec2KeyPair`

用于计算环境中启动的实例的 EC2 密钥对。您可以使用此密钥对通过 SSH 登录您的实例。

类型：字符串

必需：否

`instanceRole`

要附加到计算环境中 Amazon EC2 实例的 Amazon ECS 实例配置文件。您可以为实例配置文件指定短名称或完整的 Amazon 资源名称 (ARN)。例如，`ecsInstanceRole` 或 `arn:aws:iam::aws_account_id:instance-profile/ecsInstanceRole`。有关更多信息，请参阅Amazon ECS Instance Role (p. 86)。

类型：字符串

必需：是

`tags`

要应用于计算环境中启动的实例的键/值对标签。例如，您可以指定 `"Name": "AWS Batch Instance - C4OnDemand"` 作为标签，以便计算环境中的每个实例均具有此名称。这对于在 Amazon EC2 控制台中识别您的 AWS Batch 实例很有用。

类型：字符串到字符串映射

必需：否

`bidPercentage`

在启动实例之前，与该实例类型的按需价格进行比较时 Spot 实例价格可以达到的最大百分比。例如，如果最大百分比为 20%，则 Spot 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。

必需：否

`spotIamFleetRole`

应用于 `SPOT` 计算环境的 Amazon EC2 Spot 队组 IAM 角色的 Amazon 资源名称 (ARN)。有关更多信息，请参阅Amazon EC2 Spot Fleet Role (p. 86)。

Important

要在创建时标记 Spot 实例，此处指定的 Spot 队组 IAM 角色必须使用较新的 AmazonEC2SpotFleetTaggingRole 托管策略。以前推荐的 AmazonEC2SpotFleetRole 托管策略不具有标记 Spot 实例所需的权限。有关更多信息，请参阅在创建时未标记的 Spot 实例 (p. 115)。

类型：字符串

必需：此参数对于 `SPOT` 计算环境是必需的。

`launchTemplate`

要与计算资源关联的可选启动模板。要使用启动模板，您必须在请求中指定启动模板 ID 或启动模板名称，但不能同时指定两者。有关更多信息，请参阅 启动模板支持 (p. 59)。

类型：LaunchTemplateSpecification

object

必需：否

`launchTemplateId`

启动模板的 ID。

类型：字符串

必需：否

`launchTemplateName`

启动模板的名称。

类型：字符串

必需：否

`version`

启动模板的版本号。

类型：字符串

必需：否

# 服务角色

`serviceRole`

允许 AWS Batch 代表您调用其他 AWS 服务的 IAM 角色的完整 Amazon 资源名称 (ARN)。有关更多信息，请参阅AWS Batch Service IAM Role (p. 83)。

类型：字符串

必需：是

# 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。

`BEST_FIT`

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，或者用户达到 Amazon EC2 服务限额，则在当前正在运行的作业完成之前，不会运行其他作业。此分配策略可降低成本，但会限制扩展。

`BEST_FIT_PROGRESSIVE`

AWS Batch 将选择足够大的、能够满足队列中作业要求的额外实例类型，并优先选择每单位 vCPU 成本更低的实例类型。如果以前选择的实例类型没有可用的额外实例，AWS Batch 将选择新的实例类型。

`SPOT_CAPACITY_OPTIMIZED`

AWS Batch 将选择一个或多个足够大的、能够满足队列中作业要求的实例类型，并优先选择不太可能会中断的实例类型。此分配策略仅适用于 Spot 实例计算资源。

同时使用 `BEST_FIT_PROGRESSIVE` 和 `SPOT_CAPACITY_OPTIMIZED` 策略时，AWS Batch 可能需要超过 `maxvCpus` 以满足您的容量要求。在这种情况下，AWS Batch 永远不会超过 `maxvCpus` 一个实例。

# 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时，该代理必须确定计算资源可为作业保留多少内存。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 实例所标示的已安装内存量。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 内存对作业可用。

如果为作业指定 8192 MiB;，而您的任何一个计算资源都没有 8192 MiB 或更高内存可用于满足此需求，则作业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。

默认的 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 (p. 72)。

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
```

运行 Amazon ECS-optimized 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
```

此实例具有 8373026816 字节的总内存，这表示有 7985MiB 内存可用于任务。

# 预留系统内存

如果作业占用了计算资源上的全部内存，则作业可能与关键系统进程争用内存，并且可能会触发系统故障。Amazon ECS 容器代理提供名为 `ECS_RESERVED_MEMORY` 的配置变量，您可以使用该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。

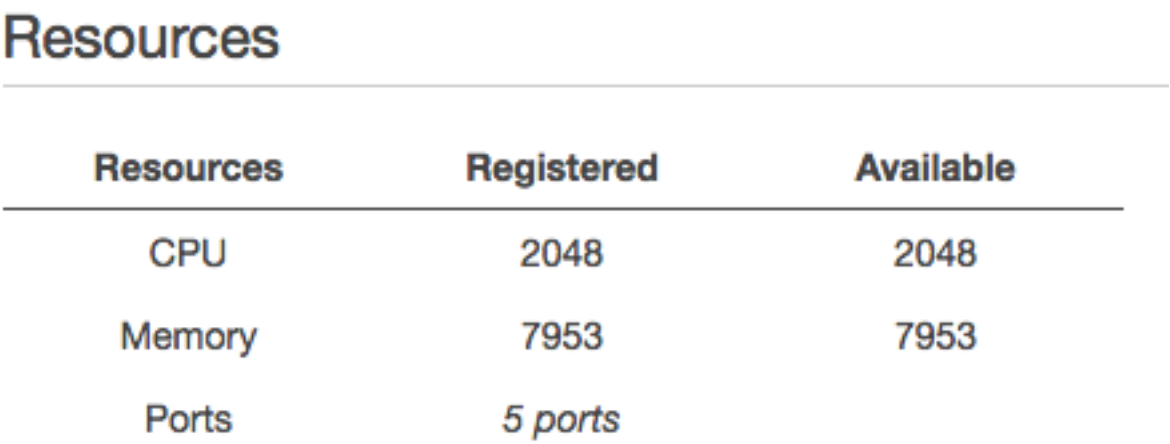
默认的 AWS Batch 计算资源 AMI 会为 Amazon ECS 容器代理以及其他关键系统进程预留 32 MiB 内存。

# 查看计算资源内存

您可以在 Amazon ECS 控制台中（或者使用 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/ 上打开 Amazon ECS 控制台。
2. 选择托管您的计算资源的集群来查看。 计算环境的集群名称以该计算环境名称开头。
3. 选择 ECS Instances (ECS 实例)，然后从 Container Instance (容器实例) 列中选择要查看的计算资源。
4. Resources (资源) 部分显示计算资源的已注册内存和可用内存。




Registered (已注册) 内存值是计算资源在首次启动时注册到 Amazon ECS 的内存量；而 Available (可用) 内存值是尚未分配给作业的内存量。

# 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。

- 计算环境仅包含受支持的实例类型（`c5n.18xlarge`、`c5n.metal`、`i3en.24xlarge`、`m5dn.24xlarge`、`m5n.24xlarge`、`r5dn.24xlarge`、`r5n.` 和 `p3dn.24xlarge`）。
- AMI 中支持 EFA 的操作系统：Amazon Linux、Amazon Linux 2、Red Hat Enterprise Linux 7.6、CentOS 7.6、Ubuntu 16.04 和 Ubuntu 18.04。
- AMI 中加载了 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 的所有实例都应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。以下示例演示了具体的区别，但没有提供必需值。

  Example 多节点并行作业的示例

```
{
  "jobDefinitionName": "EFA-MNP-JobDef",
  "type": "multinode",
  "nodeProperties": {
    ...
    "nodeRangeProperties": [
      {
        ...
        "container": {
          ...
          "linuxParameters": {
            "devices": [
              {
                "hostPath": "/dev/infiniband/uverbs0",
                "containerPath": "/dev/infiniband/uverbs0",
                "permissions": [
                    "READ", "WRITE", "MKNOD"
                ]
              },
            ],
          },
        },
      },
    ],
  },
}
```

  Example 单节点容器作业的示例

```
{
  "jobDefinitionName": "EFA-Container-JobDef",
  "type": "container",
  ...
```

```
  "containerProperties": {
    ...
    "linuxParameters": {
      "devices": [
        {
          "hostPath": "/dev/infiniband/uverbs0",
        },
      ],
    },
  },
}
```

有关 EFA 的更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Elastic Fabric Adapter。

# AWS Batch IAM Policies, Roles, and Permissions

默认情况下，IAM 用户无权创建或修改 AWS Batch 资源或使用 AWS Batch API 执行任务。这意味着，这些用户也无法使用 AWS Batch 控制台或 AWS CLI 执行这些操作。允许 IAM 用户创建或修改资源并提交职位，您必须创建 IAM 授予的政策 IAM 用户允许使用他们所需的特定资源和API操作。然后，将这些策略附加到需要这些权限的 IAM 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使用指定资源执行指定任务。有关详细信息，请参阅 权限和策略 在 IAM 用户指南。有关管理和创建自定义的更多信息 IAM 政策，请参阅 管理 IAM 政策.

同样，AWS Batch 将代表您调用其他 AWS 服务，因此服务必须使用您的凭证进行身份验证。此身份验证通过以下方式完成：创建可提供这些权限的 IAM 角色和策略，然后在创建计算环境时将该角色与计算环境关联。有关详细信息，请参阅 Amazon ECS Instance Role (p. 86)， IAM 角色， 使用服务链接的角色，和 创建角色以授权对AWS服务的权限 在 IAM 用户指南.

Getting Started

IAM 策略必须授予或拒绝使用一个或多个 AWS Batch 操作的权限。

主题



## Policy Structure

以下主题说明 IAM 策略的结构。

主题



### Policy Syntax

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"
        }
      }
    }
  ]
}
```

组成语句的各个元素如下：

- Effect: The effect can be `Allow` or `Deny`. By default, IAM users don't have permission to use resources and API actions, so all requests are denied. An explicit allow overrides the default. An explicit deny overrides any allows.
- Action: The action is the specific API action for which you are granting or denying permission. To learn about specifying action, see Actions for AWS Batch (p. 76).
- Resource: The resource that's affected by the action. Some AWS Batch API actions allow you to include specific resources in your policy that can be created or modified by the action. To specify a resource in the statement, use its Amazon Resource Name (ARN). For more information, see Supported Resource-Level Permissions for AWS Batch API Actions (p. 77) and Amazon Resource Names for AWS Batch (p. 76). If the AWS Batch API operation currently does not support resource-level permissions, you must use the * wildcard to specify that all resources can be affected by the action.
- Condition: Conditions are optional. They can be used to control when your policy is in effect.

有关 AWS Batch 的示例 IAM 策略语句的详细信息，请参阅Creating AWS Batch IAM Policies (p. 83)。

# Actions for 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，使用API操作名称的以下前缀: `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，如下所示：

```
"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，如下所示：

```
"Action": "batch:Describe*"
```

要指定所有 AWS Batch API 操作，请使用 * 通配符，如下所示：

```
"Action": "batch:*"
```

有关 AWS Batch 操作的列表，请参阅 https://docs.aws.amazon.com/batch/latest/APIReference/API_Operations.htmlAWS Batch API 参考 中的操作。

# Amazon Resource Names for AWS Batch

每个 IAM 策略语句适用于您使用资源的 ARN 指定的资源。

ARN 具有以下一般语法。

```
arn:aws:[service]:[region]:[account]:resourceType/resourcePath
```

service

服务 (例如，batch)。

region

资源所在区域 (例如，us-east-2)。

account

AWS 账户 ID，不包含连字符 (例如，123456789012)。

resourceType

资源类型 (例如，compute-environment)。

resourcePath

识别资源的路径。您可以在路径中使用 * 通配符。

AWS Batch API 操作当前支持多个 API 操作的资源级权限。有关更多信息，请参阅 Supported Resource-Level Permissions for AWS Batch API Actions (p. 77)。) 要指定所有资源，或者如果特定 API 操作不支持 ARN，请在 Resource 元素中使用 * 通配符，如下所示：

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

## Checking That Users Have the Required Permissions

在实施 IAM 策略之前，建议您检查它是否允许用户使用其所需的特定 API 操作和资源。

首先，创建一个用于测试目的的 IAM 用户，然后将 IAM 策略附加到该测试用户。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 策略模拟器测试您的策略。有关PolicySimulator的详细信息，请参阅 与 IAM 策略模拟器 在 IAM 用户指南.

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。有关详细信息，请参阅 解码授权消息 在 AWS Security Token Service API Reference，和 解码授权消息 在 AWS CLI Command Reference.

# Supported Resource-Level Permissions for AWS Batch API Actions

术语 resource-level permissions 指指定允许用户执行操作的资源的能力。 AWS Batch 对资源级别权限具有部分支持。对于某些 AWS Batch 操作，您可以控制何时允许用户执行操作（基于必须满足的条件）或是允

许用户使用的特定资源。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。

以下列表描述 AWS Batch 当前支持资源级别权限的API操作以及每个操作的支持资源、资源ARP和条件密钥。

Important

如果 AWS Batch API操作未列于此列表中，然后它不支持资源级别权限。如果 AWS Batch API 操作不支持资源级权限，则您可以向用户授予使用该操作的权限，但是必须为策略语句的资源元素指定 * 通配符。

CreateComputeEnvironment

创建 AWS Batch 计算环境。

Resource

Compute Environment

arn:aws::批次:*region*:*account*计算环境*compute-environment-name*

Condition keys

不适用*

CreateJobQueue

创建 AWS Batch 职位队列。

Resource

Compute Environment

arn:aws::批次:*region*:*account*计算环境*compute-environment-name*

Job Queue

arn:aws:批次:*region*:*account*作业队列*queue-name*

Condition keys

不适用*

DeleteComputeEnvironment

删除 AWS Batch 计算环境。

Resource

Compute Environment

arn:aws::批次:*region*:*account*计算环境*compute-environment-name*

Condition keys

不适用*

DeleteJobQueue

删除指定的作业队列。

Resource

Job Queue

arn:aws:批次:*region*:*account*作业队列*queue-name*

Condition keys

不适用*

DeregisterJobDefinition

Deregisters AWS Batch 职位定义。

Resource

Job Definition

arn:aws:批次:*region*:*account*:工作定义/*definition-name*:*revision*

Condition keys

不适用*

RegisterJobDefinition

注册 AWS Batch 定义。

Resource

Job Definition

arn:aws:批次:*region*:*account*:工作定义/*definition-name*:*revision*

Condition keys

batch:User

batch:Privileged

batch:Image

SubmitJob

提交 AWS Batch 工作定义中的工作。

Resource

Job Definition

arn:aws:批次:*region*:*account*:工作定义/*definition-name*:*revision*

Job Queue

arn:aws:批次:*region*:*account*作业队列*queue-name*

Condition keys

不适用*

UpdateComputeEnvironment

更新 AWS Batch 计算环境。

Resource

Compute Environment

arn:aws::批次:*region*:*account*计算环境*compute-environment-name*

Condition keys

不适用*

UpdateJobQueue

更新作业队列。

Resource

Job Queue

arn:aws:批次:*region*:*account*作业队列*queue-name*

Condition keys

不适用*

# Example Policies

以下示例显示了您可用于控制 IAM 用户 AWS Batch 权限的策略语句。

示例：



## Example: Read-Only Access

以下策略向用户授予使用名称以 `Describe` 和 `List` 开头的所有 AWS Batch API 操作的权限。

用户无权对资源执行任何操作 (除非其他语句为用户授予执行此操作的权限)，因为在默认情况下会对用户拒绝使用 API 操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:Describe*",
                "batch:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

## Example: Restricting to POSIX User, Docker Image, Privilege Level, and Role on Job Submission

以下策略允许用户管理自己的一组受限作业定义。

第一个和第二个报表允许用户注册并删除名称预固定的任何职位定义名称 *`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。有关详细信息，请参阅 注册工作定义 在 AWS Batch API 参考。在此示例中，只有当POSIX用户设置为 `nobody`，特权标志设置为 `false`，图像设置为 `myImage` 在 Amazon ECR 资料库。

> **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 策略中使用直接 `uid` 值来避免此名称解析。AWS Batch API 和 `batch:User` IAM 条件键都支持数字值。

第三个语句限制用户仅将特定角色传递给作业定义。

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

```
            "Effect": "Allow",
            "Action": [
                "batch:RegisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/JobDefA_*"
            ],
            "Condition": {
                "StringEquals": {
                    "batch:User": [
                        "nobody"
                    ],
                    "batch:Image": [
                        "<aws_account_id>.dkr.ecr.<aws_region>.amazonaws.com/myImage"
                    ]
                },
                "Bool": {
                    "batch:Privileged": "false"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "batch:DeregisterJobDefinition"
            ],
            "Resource": [
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/JobDefA_*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<aws_account_id>:role/MyBatchJobRole"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "batch.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

# Example: Restrict to Job Definition Prefix on Job Submission

以下策略允许用户向任何职位队列提交任何职位队列，其中任何职位定义名称以 *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。

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

```
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": [
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/JobDefA_*",
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-queue/*"
            ]
        }
    ]
}
```

## Example: Restrict to Job Queue

以下策略允许用户将职位提交至特定职位队列，名为 queue1，任何职位定义名称。

> Important
>
> 在限定作业提交的资源级访问时，必须同时提供作业队列和作业定义资源类型。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:SubmitJob"
            ],
            "Resource": [
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-definition/*",
                "arn:aws:batch:<aws_region>:<aws_account_id>:job-queue/queue1"
            ]
        }
    ]
}
```

# AWS Batch 托管策略

AWS Batch 提供了一个托管策略，可将该策略附加到 IAM 用户来提供使用 AWS Batch 资源和 API 操作的权限。您可以直接应用此策略，也可以以它为起点创建自己的策略。有关这些策略中提到的每个 API 操作的更多信息，请参阅 AWS Batch API 参考 中的操作。

## AWSBatchFullAccess

此策略授予对 AWS Batch 的完全管理员访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "batch:*",
                "cloudwatch:GetMetricStatistics",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeVpcs",
```

```
                "ec2:DescribeImages",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ecs:DescribeClusters",
                "ecs:Describe*",
                "ecs:List*",
                "logs:Describe*",
                "logs:Get*",
                "logs:TestMetricFilter",
                "logs:FilterLogEvents",
                "iam:ListInstanceProfiles",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/AWSBatchServiceRole",
                "arn:aws:iam::*:role/service-role/AWSBatchServiceRole",
                "arn:aws:iam::*:role/ecsInstanceRole",
                "arn:aws:iam::*:instance-profile/ecsInstanceRole",
                "arn:aws:iam::*:role/iaws-ec2-spot-fleet-role",
                "arn:aws:iam::*:role/aws-ec2-spot-fleet-role",
                "arn:aws:iam::*:role/AWSBatchJobRole*"
            ]
        }
    ]
}
```

# Creating AWS Batch IAM Policies

可以创建特定的 IAM 策略来限制您账户中的用户有权访问的调用和资源，然后将这些策略与 IAM 用户关联。

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。有关详细信息，请参阅 权限和策略 在 IAM 用户指南。有关管理和创建自定义的更多信息 IAM 政策，请参阅 管理 IAM 政策.

# AWS Batch Service IAM Role

AWS Batch 代表您调用其他 AWS 服务以管理服务所使用的资源。您必须先具有向 AWS Batch 提供必需权限的 IAM 策略和角色，然后才能使用服务。

大多数情况下，在控制台首次运行体验中将自动为您创建 AWS Batch 服务角色。您可使用以下过程来检查您的账户是否已具有 AWS Batch 服务角色。

`AWSBatchServiceRole` 策略如下所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeInstances",
```

```
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeImages",
                "ec2:DescribeImageAttribute",
                "ec2:DescribeSpotInstanceRequests",
                "ec2:DescribeSpotFleetInstances",
                "ec2:DescribeSpotFleetRequests",
                "ec2:DescribeSpotPriceHistory",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:CreateLaunchTemplate",
                "ec2:DeleteLaunchTemplate",
                "ec2:RequestSpotFleet",
                "ec2:CancelSpotFleetRequests",
                "ec2:ModifySpotFleetRequest",
                "ec2:TerminateInstances",
                "ec2:RunInstances",
                "autoscaling:DescribeAccountLimits",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeLaunchConfigurations",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:CreateLaunchConfiguration",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:SetDesiredCapacity",
                "autoscaling:DeleteLaunchConfiguration",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:SuspendProcesses",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:TerminateInstanceInAutoScalingGroup",
                "ecs:DescribeClusters",
                "ecs:DescribeContainerInstances",
                "ecs:DescribeTaskDefinition",
                "ecs:DescribeTasks",
                "ecs:ListClusters",
                "ecs:ListContainerInstances",
                "ecs:ListTaskDefinitionFamilies",
                "ecs:ListTaskDefinitions",
                "ecs:ListTasks",
                "ecs:CreateCluster",
                "ecs:DeleteCluster",
                "ecs:RegisterTaskDefinition",
                "ecs:DeregisterTaskDefinition",
                "ecs:RunTask",
                "ecs:StartTask",
                "ecs:StopTask",
                "ecs:UpdateContainerAgent",
                "ecs:DeregisterContainerInstance",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "iam:GetInstanceProfile",
                "iam:GetRole"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "*"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "ecs-tasks.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com",
                        "autoscaling.amazonaws.com",
                        "ecs.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "RunInstances"
                }
            }
        }
    ]
}
```

您可以使用以下过程检查您的账户是否已拥有 AWS Batch 服务角色并附加托管 IAM 策略（如果需要）。

### 在 IAM 控制台中检查 **AWSBatchServiceRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 搜索角色列表 `AWSBatchServiceRole`。如果角色不存在，请使用以下步骤创建角色。如果角色存在，请选择角色以查看附加的策略。
4. 选择 Permissions。
5. 确保将 AWSBatchServiceRole 托管策略附加到该角色。如果附加该策略，则将正确配置 AWS Batch 服务角色。否则，请执行以下子步骤来附加策略。
   a. 选择 Attach Policy (附加策略)。
   b. 要缩小要附加的可用策略的列表范围，请为 Filter (筛选条件) 键入 AWSBatchServiceRole。
   c. 选择 AWSBatchServiceRole 策略，然后选择 Attach Policy。
6. 选择 Trust Relationships，然后选择 Edit Trust Relationship。
7. 验证信任关系是否包含以下策略。如果信任关系符合以下策略，请选择 Cancel。如果信任关系不符合，请将策略复制到 Policy Document 窗口中并选择 Update Trust Policy。

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

### 创建 **AWSBatchServiceRole** IAM 角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles 和 Create New Role。
3. 对于 Select type of trusted entity (选择受信任实体的类型)，选择 AWS service (AWS 服务)。对于 Choose the service that will use this role (选择将使用此角色的服务)，选择 Batch (批处理)。
4. 选择 Next (下一步)。权限， 下一步: 标签，和 下一步: 审核
5. 对于 Role Name (角色名称)，键入 `AWSBatchServiceRole`，然后选择 Create Role (创建角色)。

# Amazon ECS Instance Role

AWS Batch 计算环境中将使用 Amazon ECS 容器实例填充，并且将在本地运行 Amazon ECS 容器代理。Amazon ECS 容器代理将代表您调用各种 AWS API 操作，因此，运行代理的容器实例需要这些服务的 IAM 策略和角色，以便了解属于您的代理。您必须先为这些容器实例创建它们启动时要使用的 IAM 角色和实例配置文件，然后才能创建计算环境并在该环境中启动容器实例。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。

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，您可以使用以下过程检查并确定您的账户是否已拥有 Amazon ECS 实例角色和实例配置文件并附加托管 IAM 策略（如果需要）。

### 在 IAM 控制台中检查 **ecsInstanceRole**

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 搜索角色列表 `ecsInstanceRole`。如果角色不存在，请使用以下步骤创建角色。
    a. 选择 Create Role。
    b. 对于 Select type of trusted entity (选择受信任实体的类型)，选择 AWS service (AWS 服务)。对于 Choose the service that will use this role (选择将使用此角色的服务)，选择 Elastic Container Service。对于选择您的使用案例，选择 EC2 Role for Elastic Container Service (Elastic Container Service 的 EC2 角色)。
    c. 选择 Next (下一步)。权限， 下一步: 标签，和 下一步: 审核
    d. 对于 Role Name (角色名称)，键入 `ecsInstanceRole`，然后选择 Create Role (创建角色)。

# Amazon EC2 Spot Fleet Role

如果您创建一个使用 Amazon EC2 Spot 队列实例的托管计算环境，则必须创建一个角色，该角色应向 Spot 队列授予代表您启动、标记和终止实例的权限。在 Spot 队列请求中指定该角色。您还必须具有适用于 Amazon EC2 Spot 和 Spot 队组的 AWSServiceRoleForEC2Spot 和 AWSServiceRoleForEC2SpotFleet 服

务相关角色。使用以下过程创建所有这些角色。有关详细信息，请参阅 使用服务链接的角色 和 创建角色以授权对AWS服务的权限 在 IAM 用户指南.

主题



# Create Amazon EC2 Spot Fleet Roles in the AWS 管理控制台

创建适用于 Amazon EC2 Spot 的 **`AWSServiceRoleForEC2Spot`** IAM 服务相关角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles 和 Create role。
3. 对于 Select type of trusted entity (选择受信任实体的类型)，选择 AWS service (AWS 服务)。对于 Choose the service that will use this role (选择将使用此角色的服务)，选择 EC2。
4. 在选择您的使用案例部分中，选择 EC2 - Spot Instances (EC2 - Spot 实例)。
5. 选择 Next (下一步)。权限， 下一步: 标签，和 下一步: 审核
6. 对于 Role Name (角色名称)，键入 `AmazonEC2SpotFleetRole`。选择 Create Role (创建角色)。

创建适用于 Amazon EC2 Spot 队组的 **`AWSServiceRoleForEC2SpotFleet`** IAM 服务相关角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Roles 和 Create role。
3. 对于 Select type of trusted entity (选择受信任实体的类型)，选择 AWS service (AWS 服务)。对于 Choose the service that will use this role (选择将使用此角色的服务)，选择 EC2。
4. 在选择您的使用案例部分中，选择 EC2 - Spot Fleet (EC2 - Spot 队列)。
5. 选择 Next (下一步)。权限， 下一步: 标签，和 下一步: 审核
6. 对于 角色名称，类型 `AWSServiceRoleForEC2SpotFleet`。选择 创建角色.

   Note

   一直以来，Amazon EC2 Spot 队列角色都有两种托管策略。

   - AmazonEC2SpotFleetRole: This was the original managed policy for the Spot Fleet role. It has tighter IAM permissions, but it does not support Spot Instance tagging in compute environments. If you've previously created a Spot Fleet role with this policy, see 在创建时未标记的 Spot 实例 (p. 115) to apply the new recommended policy to that role.
   - AmazonEC2SpotFleetTaggingRole: This role provides all of the necessary permissions to tag Amazon EC2 Spot Instances. Use this role to allow Spot Instance tagging on your AWS Batch compute environments.

# Create Amazon EC2 Spot Fleet Roles with the AWS CLI

为 Spot 队组计算环境创建 **`AmazonEC2SpotFleetRole`** IAM 角色

1. 使用 AWS CLI 运行以下命令：

```
aws iam create-role --role-name AmazonEC2SpotFleetRole \
    --assume-role-policy-document '{"Version":"2012-10-17","Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"spotfleet.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

2. 要将 `AmazonEC2SpotFleetTaggingRole` 托管的 IAM 策略附加到您的 `AmazonEC2SpotFleetRole` 角色，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
```

创建适用于 Amazon EC2 Spot 的 **`AWSServiceRoleForEC2Spot`** IAM 服务相关角色

- 使用 AWS CLI 运行以下命令：

```
aws iam create-service-linked-role --aws-service-name spot.amazonaws.com
```

创建适用于 Amazon EC2 Spot 队组的 **`AWSServiceRoleForEC2SpotFleet`** IAM 服务相关角色

- 使用 AWS CLI 运行以下命令：

```
aws iam create-service-linked-role --aws-service-name spotfleet.amazonaws.com
```

# CloudWatch Events IAM Role

Amazon CloudWatch Events 提供近乎实时的系统事件流，这些系统事件描述 Amazon Web Services 资源的变化。AWS Batch 作业可作为 CloudWatch Events 目标提供。通过使用可快速设置的简单规则，您可以匹配事件并提交 AWS Batch 任务以响应这些事件。CloudWatch Events 必须获得代表您运行 AWS Batch 作业的权限，然后您才能提交包含 CloudWatch Events 规则和目标的 AWS Batch 作业。

> **Note**
>
> 当您在 CloudWatch Events 控制台中创建将 AWS Batch 队列指定为目标的规则时，您将获得创建此角色的机会。有关示例演练的信息，请参阅作为 CloudWatch Events 目标的 AWS Batch 任务 (p. 91)。

您的 CloudWatch Events IAM 角色的信任关系必须为 `events.amazonaws.com` 服务委托人提供代入该角色的能力，如下所示。

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

附加到您的 CloudWatch Events IAM 角色的策略应允许对您的资源的 `batch:SubmitJob` 权限。AWS Batch 提供了 `AWSBatchServiceEventTargetRole` 以提供这些权限，如下所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "batch:SubmitJob"
       ],
      "Resource": "*"
    }
  ]
}
```

# CloudWatch Events 的 AWS Batch 事件流

您可以使用 CloudWatch Events 的 AWS Batch 事件流近乎实时地接收通知，了解已提交到任务队列的任务的当前状态。

利用 CloudWatch Events，您可以监控任务的进度，构建包含复杂依赖关系的 AWS Batch 自定义工作流程，生成使用率报告或有关任务执行的指标，或构建您自己的自定义控制面板。借助 AWS Batch 和 CloudWatch Events，您无需计划和监控用于持续轮询 AWS Batch 以了解任务状态更改的代码，而是使用任何 CloudWatch Events 目标（如 AWS Lambda、Amazon Simple Queue Service、Amazon Simple Notification Service 或 Amazon Kinesis Data Streams）以异步方式处理 AWS Batch 任务状态更改。

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，事件提供了足够的信息来识别重复情况 (您可以比较事件的时间戳和作业状态)。

AWS Batch 任务可作为 CloudWatch Events 目标提供。通过使用可快速设置的简单规则，您可以匹配事件并提交 AWS Batch 任务以响应这些事件。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的什么是 Amazon CloudWatch Events？。您还可以使用 CloudWatch Events 来计划使用 cron 或 rate 表达式在某些时间自行触发的自动化操作。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的规则的计划表达式。有关示例演练的信息，请参阅 作为 CloudWatch Events 目标的 AWS Batch 任务 (p. 91)。

主题



# AWS Batch 事件

AWS Batch 将任务状态更改事件发送到 CloudWatch Events。AWS Batch 跟踪任务的状态。如果之前提交的作业的状态发生更改，将触发事件，例如 `RUNNING` 状态的作业进入 `FAILED` 状态。这些事件归类为作业状态更改事件。

> **Note**
>
> AWS Batch 将来可能会增加其他事件类型、源和详细信息。如果您以编程方式对事件 JSON 数据反序列化，请确保应用程序已准备好处理未知属性，以避免在增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，就会创建事件。有关 AWS Batch 任务状态的更多信息，请参阅作业状态 (p. 14)。

> **Note**
>
> 对于初始作业提交不会创建事件。

Example 作业状态更改事件

任务状态更改事件以下面的形式传送（下面的 `detail` 部分类似于从 AWS Batch API 参考 中的 DescribeJobs API 操作返回的 Job 对象）。有关 CloudWatch Events 参数的更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的事件和事件模式。

```
{
  "version": "0",
  "id": "c8f9c4b5-76e5-d76a-f980-7011e206042b",
  "detail-type": "Batch Job State Change",
  "source": "aws.batch",
  "account": "aws_account_id",
  "time": "2017-10-23T17:56:03Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:batch:us-east-1:aws_account_id:job/4c7599ae-0a82-49aa-ba5a-4727fcce14a8"
  ],
  "detail": {
    "jobName": "event-test",
    "jobId": "4c7599ae-0a82-49aa-ba5a-4727fcce14a8",
    "jobQueue": "arn:aws:batch:us-east-1:aws_account_id:job-queue/HighPriority",
    "status": "RUNNABLE",
    "attempts": [],
    "createdAt": 1508781340401,
    "retryStrategy": {
      "attempts": 1
    },
    "dependsOn": [],
    "jobDefinition": "arn:aws:batch:us-east-1:aws_account_id:job-definition/first-run-job-
definition:1",
    "parameters": {},
    "container": {
      "image": "busybox",
      "vcpus": 2,
      "memory": 2000,
      "command": [
        "echo",
        "'hello world'"
      ],
      "volumes": [],
      "environment": [],
      "mountPoints": [],
      "ulimits": []
    }
  }
}
```

# 作为 CloudWatch Events 目标的 AWS Batch 任务

Amazon CloudWatch Events 提供近乎实时的系统事件流，这些系统事件描述 Amazon Web Services 资源的变化。AWS Batch 任务可作为 CloudWatch Events 目标提供。通过使用可快速设置的简单规则，您可以匹配事件并提交 AWS Batch 任务以响应这些事件。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的什么是 Amazon CloudWatch Events？。

您还可以使用 CloudWatch Events 来计划使用 cron 或 rate 表达式在某些时间自行触发的自动化操作。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的规则的计划表达式。

AWS Batch 任务充当 CloudWatch Events 目标的一些常用案例为：

- 创建定期执行的计划任务，例如在低使用率时段内（此时 Amazon EC2 Spot 实例的价格较低）执行的 cron 任务。
- 运行 AWS Batch 任务以响应在 CloudTrail 中记录的 API 操作，例如，在对象上传到指定 Amazon S3 存储桶时自动提交任务，然后使用 CloudWatch Events 输入转换器将存储桶和对象的键名称传递至 AWS Batch 参数。

Note

在这种情况下，所有 AWS 资源（Amazon S3 存储桶、CloudWatch Events 规则、CloudTrail 日志等）位于同一区域。

CloudWatch Events 服务需要获得代表您运行 AWS Batch 任务的权限，然后您才能提交包含 CloudWatch Events 规则和目标的 AWS Batch 任务。当您在 CloudWatch Events 控制台中创建将 AWS Batch 任务指定为目标的规则时，您将获得创建此角色的机会。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 CloudWatch Events IAM Role (p. 88)。

# 创建计划 AWS Batch 任务

以下过程现实了如何创建计划 AWS Batch 任务和所需的 CloudWatch Events IAM 角色。

使用 CloudWatch Events 创建计划 AWS Batch 任务

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中，依次选择 Events (事件) 和 Create rule (创建规则)。
3. 对于 Event source (事件源)，选择 Schedule (计划)，然后选择是对计划规则使用固定间隔计划还是 `cron` 表达式。有关更多信息，请参阅 Amazon CloudWatch Events 用户指南 中的规则的计划表达式。
   - 对于 Fixed rate of (固定频率为)，为计划输入时间间隔和单位。
   - 对于 Cron expression，为任务计划输入 `cron` 表达式。这些表达式有六个必填字段，字段以空格分隔。有关更多信息及 `cron` 表达式的示例，请参阅 Amazon CloudWatch Events 用户指南 中的 Cron 表达式。
4. 对于 Targets，选择 Add target。
5. 选择 Batch job queue (批处理任务队列) 并相应地填写以下字段：
   - Job queue (任务队列)：输入您在其中计划任务的任务队列的 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Array Jobs (p. 17)。
   - 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 (p. 16)。
6. 选择要用于任务的现有 CloudWatch Events IAM 角色，或选择 Create a new role for this specific resource (为此特定资源创建新角色) 以创建新角色。有关更多信息，请参阅CloudWatch Events IAM Role (p. 88)。
7. 对于 Rule definition (规则定义)，相应地填写以下字段，然后选择 Create rule (创建规则)。
   - Name (名称)：输入规则的名称。
   - Description (描述)：(可选) 输入规则的描述。
   - State (状态)：选择是启用规则以便规则在下一个时间间隔开始计划，还是禁用规则直到某个将来的日期。

# 使用 CloudWatch Events 输入转换器将事件信息传递至 AWS Batch 目标

您可以使用 CloudWatch Events 输入转换器在任务提交时将事件信息传递至 AWS Batch。这在您因其他 AWS 事件信息触发任务（如将对象上传至 Amazon S3 存储桶）的情况下可能尤为重要。您可以将作业定义

与容器命令中的参数替代值一起使用，而且 CloudWatch Events 输入转换器可以基于事件数据提供参数值。例如，以下任务定义应看到名为 *S3bucket* 和 *S3key* 的参数值。

```
{
  "jobDefinitionName": "echo-parameters",
  "containerProperties": {
    "image": "busybox",
    "vcpus": 2,
    "memory": 2000,
    "command": [
      "echo",
      "Ref::S3bucket",
      "Ref::S3key"
    ]
  }
}
```

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（Amazon S3 存储桶、CloudWatch Events 规则、CloudTrail 日志等）位于同一区域。

### 创建使用输入转换器的 AWS Batch 目标

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧导航窗格中，依次选择 Events (事件) 和 Create rule (创建规则)。
3. 对于 Event source (事件源)，选择 Event Pattern (事件模式)，然后根据需要构建规则以满足应用程序需求。
4. 对于 Targets (目标)，选择 Batch job queue (批处理作业队列)，然后指定要用于由此规则触发的作业的作业队列、作业定义和作业名称。
5. 选择 Configure input (配置输入)，然后选择 Input Transformer (输入转换器)。
6. 在上方输入转换器文本框中，指定要从触发事件中解析的值。例如，要解析 Amazon S3 事件中的存储桶和键名称，请使用以下 JSON。

   ```
   {"S3BucketValue":"$.detail.requestParameters.bucketName","S3KeyValue":"$.detail.requestParameters.k
   ```

7. 在下方输入转换器文本框中，创建要传递至 AWS Batch 任务的 `Parameters` 结构。这些参数将被替换为作业运行时作业容器的命令中的 *Ref::S3bucket* 和 *Ref::S3key* 占位符。

   ```
   {"Parameters" : {"S3bucket": <S3BucketValue>, "S3key": <S3KeyValue>}}
   ```

8. 选择要用于任务的现有 CloudWatch Events IAM 角色，或选择 Create a new role for this specific resource (为此特定资源创建新角色) 以创建新角色。有关更多信息，请参阅CloudWatch Events IAM Role (p. 88)。
9. 选择 Configure details (配置详细信息)，然后对于 Rule definition (规则定义)，相应地填写以下字段，然后选择 Create rule (创建规则)。
   - Name (名称)：输入规则的名称。
   - Description (描述)：(可选) 输入规则的描述。
   - State (状态)：选择是立即启用规则，还是禁用规则直到某个将来的日期。

# 教程：侦听 AWS Batch CloudWatch Events

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 入门 (p. 8)中的步骤创建一个。在本教程结束时，您可向此作业队列提交作业，以测试您是否已正确配置 Lambda 函数。

## 步骤 1：创建 Lambda 函数

在此过程中，您将创建一个简单的 Lambda 函数来充当 AWS Batch 事件流消息的目标。

创建目标 Lambda 函数

1. 通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 依次选择 Create a Lambda function、Author from scratch。
3. 对于 Name，输入 batch-event-stream-handler。
4. 对于 Role，依次选择 Create a custom role 和 Allow。
5. 选择 Create function。
6. 在 Function code 部分中，为运行时选择 Python 2.7，并编辑示例代码以匹配以下示例：

```
import json

def lambda_handler(event, 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 2.7 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 Save。

## 步骤 2：注册事件规则

接下来，您创建一个 CloudWatch Events 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)，您可使用这些信息以编程方式对事件进行筛选和排序。

> Note
>
> 在使用 AWS 管理控制台创建事件规则时，控制台会自动添加所需的 IAM 权限以便向 CloudWatch Events 授予调用 Lambda 函数所需的权限。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 CloudWatch 用户指南 中的事件和事件模式。

创建您的 CloudWatch Events 规则

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格上，选择 Events 和 Create rule。

3. 对于 Event source，选择 Event Pattern 作为事件源，然后选择 Build custom event pattern。
4. 在文本区域中粘贴以下事件模式。

```
{
  "source": [
    "aws.batch"
  ]
}
```

此规则将应用于您的所有 AWS Batch 组的所有 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。

5. 对于 Targets，选择 Add target。对于 Target type (目标类型)，选择 Lambda function (Lambda 函数)，然后选择您的 Lambda 函数。
6. 选择 Configure details。
7. 对于 Rule definition，键入规则的名称和说明，然后选择 Create rule。

## 第 3 步：测试您的配置

最后，您可以通过向作业队列提交作业来测试您的 CloudWatch Events 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. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 提交新的 AWS Batch 作业。有关更多信息，请参阅提交作业 (p. 12)。
3. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
4. 在导航窗格中，选择 Logs (日志)，然后选择 Lambda 函数的日志组（例如，/aws/lambda/*my-function*）。
5. 选择日志流以查看事件数据。

# 教程：为失败的作业事件发送 Amazon Simple Notification Service 提醒

在本教程中，您将配置一个 CloudWatch Events 事件规则，它只捕获任务已转为 `FAILED` 状态的任务事件。在本教程结束时，您可向此作业队列提交作业，以测试您是否已正确配置 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 入门 (p. 8)中的步骤创建一个。

## 步骤 1：创建并订阅 Amazon SNS 主题

在本教程中，您配置一个 Amazon SNS 主题来充当新事件规则的事件目标。

创建 Amazon SNS 主题

1. 通过以下网址打开 Amazon SNS 控制台：https://console.aws.amazon.com/sns/v3/home。
2. 选择 Topics 和 Create new topic。

3. 对于Topic name (主题名称)，输入 **`JobFailedAlert`** 并选择Create topic (创建主题)。
4. 选择您刚创建的主题。在 Topic details: JobFailedAlert 屏幕上，选择 Create subscription。
5. 对于 Protocol，选择 Email。对于 Endpoint，输入您当前有权访问的电子邮件地址，然后选择 Create subscription。
6. 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 Confirm subscription。

# 步骤 2：注册事件规则

接下来，注册一个仅捕获作业失败事件的事件规则。

创建事件规则

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择 Events 和 Create rule。
3. 选择 Show advanced options 和 edit。
4. 对于 Build a pattern that selects events for processing by your targets，将现有文本替换为以下文本：

```
{
  "detail-type": [
    "Batch Job State Change"
  ],
  "source": [
    "aws.batch"
  ],
  "detail": {
    "status": [
      "FAILED"
    ]
  }
}
```

   此代码定义一个与任务状态为 `FAILED` 的任意事件匹配的 CloudWatch Events 规则。有关事件模式的更多信息，请参阅 Amazon CloudWatch 用户指南 中的事件和事件模式。
5. 对于 Targets，选择 Add target。对于 Target type，依次选择 SNS topic 和 JobFailedAlert。
6. 选择 Configure details。
7. 对于 Rule definition，键入规则的名称和说明，然后选择 Create rule。

# 步骤 3：测试您的规则

要测试您的规则，请提交一个在启动后很快就以非零退出代码退出的作业。如果您的事件规则配置正确，您将在几分钟内收到包含事件文本的电子邮件。

测试规则

1. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 提交新的 AWS Batch 作业。有关更多信息，请参阅 提交作业 (p. 12)。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电子邮件提醒。

# 使用 AWS Batch 记录 AWS CloudTrail API 调用

AWS Batch 与 AWS CloudTrail 集成，后者是一项服务，该服务提供了由用户、角色或 AWS Batch 中的 AWS 服务执行的操作的记录。CloudTrail 将 AWS Batch 的所有 API 调用作为事件捕获。捕获的调用包含来自 AWS Batch 控制台和代码的 AWS Batch API 操作调用。如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 AWS Batch 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台的 Event history (事件历史记录) 中查看最新事件。通过使用 CloudTrail 收集的信息，您可以确定向 AWS Batch 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅 AWS CloudTrail User Guide。

## CloudTrail 中的 AWS Batch 信息

在您创建 CloudTrail 账户时，即针对该账户启用了 AWS。AWS Batch 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 AWS Batch 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。此跟踪在 AWS 分区中记录来自所有区域的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 接收来自多个区域的 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 AWS Batch 操作均由 CloudTrail 记录下来并记载到 https://docs.aws.amazon.com/batch/latest/APIReference/ 中。例如，对 `SubmitJob`、`ListJobs` 和 `DescribeJobs` 部分的调用将在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员的信息。身份信息帮助您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

## 了解 AWS Batch 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 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。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateComputeEnvironment` 操作。


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2017-12-20T00:48:46Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
        "accountId": "012345678910",
        "userName": "Admin"
      }
    }
  },
  "eventTime": "2017-12-20T00:48:46Z",
  "eventSource": "batch.amazonaws.com",
  "eventName": "CreateComputeEnvironment",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "computeResources": {
      "subnets": [
        "subnet-5eda8e04"
      ],
      "tags": {
        "testBatchTags": "CLI testing CE"
      },
      "desiredvCpus": 0,
      "minvCpus": 0,
      "instanceTypes": [
        "optimal"
      ],
      "securityGroupIds": [
        "sg-aba9e8db"
      ],
      "instanceRole": "ecsInstanceRole",
      "maxvCpus": 128,
      "type": "EC2"
    },
    "state": "ENABLED",
    "type": "MANAGED",
    "serviceRole": "service-role/AWSBatchServiceRole",
    "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"
}
```

# 教程：为您的计算环境创建带有公有和私有子网的 VPC

计算环境中的计算资源需要外部网络访问权限以便与 Amazon ECS 服务终端节点进行通信。但是，您可能有想要在私有子网中运行的任务。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任务的灵活性。私有子网中的任务可以通过 NAT 网关访问 Internet。

本教程将指导您创建带有两个公有子网和两个私有子网的 VPC，这些子网获得了通过 NAT 网关访问 Internet 的功能。

## 步骤 1：为您的 NAT 网关选择弹性 IP 地址

NAT 网关要求您的公有子网中有一个弹性 IP 地址，但 VPC 向导不会为您创建该地址。在运行 VPC 向导之前创建弹性 IP 地址。

创建弹性 IP 地址

1. 打开 Amazon VPC 控制台 https://console.aws.amazon.com/vpc/。
2. 在左侧导航窗格中，选择 Elastic IPs。
3. 依次选择 Allocate new address、Allocate 和 Close。
4. 记下您新创建的弹性 IP 地址的 Allocation ID；稍后在 VPC 向导中输入此 ID。

## 步骤 2：运行 VPC 向导

VPC 向导会为您自动创建和配置您的大多数 VPC 资源。

运行 VPC 向导

1. 在左侧导航窗格中，选择 VPC Dashboard。
2. 依次选择 Launch VPC Wizard (启动 VPC 向导)、VPC with Public and Private Subnets (带有公有子网和私有子网的 VPC) 和 Select (选择)。
3. 对于 VPC name，为您的 VPC 提供一个唯一名称。
4. 对于 Elastic IP Allocation ID，选择您之前创建的弹性 IP 地址的 ID。
5. 选择 Create VPC。
6. 完成向导后，选择 OK。记下在其中创建您的 VPC 子网的可用区。您的其他子网应在不同的可用区中创建。

   非默认子网（例如通过 VPC 向导创建的子网）不是自动分配的公有 IPv4 地址。必须为在公有子网中启动的实例分配一个公有 IPv4 地址，才能与 Amazon ECS 服务终端节点通信。

修改您的公有子网的 IPv4 寻址行为

1. 在左侧导航窗格中，选择 Subnets。
2. 为您的 VPC 选择公有子网。默认情况下，通过 VPC 向导创建的名称为 Public subnet (公有子网)。
3. 依次选择 Actions (操作)、Modify auto-assign IP settings (修改自动分配 IP 设置)。

4. 选中 Enable auto-assign public IPv4 address (启用自动分配公用 IPv4 地址) 复选框，然后选择 Save (保存)。

# 步骤 3：创建额外子网

该向导将在单可用区中创建带有单个公有子网和单个私有子网的 VPC。为了获得更高的可用性，您应在另一个可用区中为每个子网类型额外创建一个，使您的 VPC 拥有跨两个可用区的公有子网和私有子网。

创建额外私有子网

1. 在左侧导航窗格中，选择 Subnets。
2. 选择 Create Subnet。
3. 对于 Name tag，为您的子网输入名称，如 Private subnet。
4. 对于 VPC，选择您之前创建的 VPC。
5. 对于 Availability Zone，选择与 VPC 中的原始子网所在的可用区不同的可用区。
6. 对于 IPv4 CIDR block，输入有效的 CIDR 块。例如，在默认情况下，该向导将在 10.0.0.0/24 和 10.0.1.0/24 中创建 CIDR 块。您可以对第二个私有子网使用 10.0.3.0/24。
7. 选择 Yes, Create。

创建额外公有子网

1. 在左侧导航窗格中，选择 Subnets，然后选择 Create Subnet。
2. 对于 Name tag，为您的子网输入名称，如 Public subnet。
3. 对于 VPC，选择您之前创建的 VPC。
4. 对于 Availability Zone，选择与您在上一个过程中创建的额外私有子网所在的可用区相同的可用区。
5. 对于 IPv4 CIDR block，输入有效的 CIDR 块。例如，在默认情况下，该向导将在 10.0.0.0/24 和 10.0.1.0/24 中创建 CIDR 块。您可以对第二个公有子网使用 10.0.2.0/24。
6. 选择 Yes, Create。
7. 选择您刚刚创建的公有子网，然后选择 Route Table、Edit。
8. 默认情况下，私有路由表处于选中状态。选择其他可用的路由表，从而将 0.0.0.0/0 目标路由至 Internet 网关 (igw-***xxxxxxxx***)，然后选择 Save。
9. 在您的第二个公有子网仍处于选中状态的情况下，选择 Subnet Actions、Modify auto-assign IP settings。
10. 依次选择 Enable auto-assign public IPv4 address、Save 和 Close。

# 后续步骤

在创建 VPC 后，您应考虑以下后续步骤：

- 如果您的公有和私有资源需要入站网络访问权限，则为这些资源创建安全组。有关更多信息，请参阅 Amazon VPC 用户指南 中的使用安全组。
- 创建一个可将计算资源启动到您的新 VPC 中的 AWS Batch 托管计算环境。有关更多信息，请参阅 创建计算环境 (p. 63)。如果您在 AWS Batch 控制台中使用了计算环境创建向导，则可以指定您刚刚创建的 VPC 以及要将您的实例启动到的公有或私有子网，具体取决于您的使用案例。
- 创建要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 (p. 48)。
- 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 Creating a Job Definition (p. 29)。
- 将带有任务定义的任务提交到您的新的任务队列。此任务将落到您使用新 VPC 和子网创建的计算环境中。有关更多信息，请参阅 提交作业 (p. 12)。

# AWS Batch 中的安全性

安全性是 AWS 和您的共同责任。责任共担模型将其描述为云的安全性和云中的安全性。

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划的一部分，第三方审计人员将定期测试和验证安全性的有效性。要了解适用于 AWS Batch 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 – 您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 AWS Batch 时应用责任共担模型。以下主题说明如何配置 AWS Batch 以实现您的安全性和合规性目标。您还将了解如何使用其他 AWS 服务来帮助您监控和保护您的 AWS Batch 资源。

主题



# 适用于 AWS Batch 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）以使用 AWS Batch 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题



## 受众

如何使用 AWS Identity and Access Management (IAM) 因您可以在 AWS Batch 中执行的操作而异。

服务用户 – 如果您使用 AWS Batch 服务来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 AWS Batch 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 AWS Batch 中的一项功能，请参阅排查 AWS Batch Identity and Access 的问题 (p. 109)。

服务管理员 – 如果您在公司负责管理 AWS Batch 资源，则您可能具有 AWS Batch 的完全访问权限。您有责任确定您的员工应访问哪些 AWS Batch 功能和资源。然后，您必须向 IAM 管理员提交请求以更改您的服务用户的权限。检查此页上的信息，了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 AWS Batch 搭配使用的更多信息，请参阅AWS Batch 如何与 IAM 协同工作 (p. 105)。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解有关您可以如何编写策略以管理 AWS Batch 访问权限的详细信息。要查看您可在 IAM 中使用的基于身份的 AWS Batch 示例策略，请参阅基于身份的策略示例 (p. 107)。

# 使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。有关使用 AWS 管理控制台登录的更多信息，请参阅 IAM 用户指南 中的 IAM 控制台和登录页面。

您必须以 AWS 账户根用户、IAM 用户身份或通过代入 IAM 角色进行身份验证（登录到 AWS）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些案例中，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其他公司的凭证访问 AWS 时，您间接地代入了角色。

要直接登录到 AWS 管理控制台，请使用您的密码和 根用户 电子邮件或 IAM 用户名。您可以使用 根用户 或 IAM 用户访问密钥以编程方式访问 AWS。AWS 提供了开发工具包和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。使用签名版本 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅 AWS General Reference 中的签名版本 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅 IAM 用户指南 中的在 AWS 中使用多重身份验证 (MFA)。

## AWS 账户根用户

当您首次创建 AWS 账户时，最初使用的是一个对账户中所有 AWS 服务和资源有完全访问权限的单点登录身份。此身份称为 AWS 账户 根用户，可使用您创建账户时所用的电子邮件地址和密码登录来获得此身份。强烈建议您不使用 根用户 执行日常任务，即使是管理任务。请遵守仅将 根用户 用于创建首个 IAM 用户的最佳实践。然后请妥善保存 根用户 凭证，仅用它们执行少数账户和服务管理任务。

## IAM 用户和群组

IAM 用户是 AWS 账户内对某个人员或应用程序具有特定权限的一个身份。IAM 用户可以拥有长期凭证，例如用户名和密码或一组访问密钥。要了解如何生成访问密钥，请参阅 IAM 用户指南 中的管理 IAM 用户的访问密钥。为 IAM 用户生成访问密钥时，请确保查看并安全保存密钥对。您以后无法找回秘密访问密钥，而是必须生成新的访问密钥对。

IAM 组 是指定一个 IAM 用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您有一个名为 IAMAdmins 的组并为该组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南 中的何时创建 IAM 用户（而不是角色）。

## IAM 角色

IAM 角色 是 AWS 账户中具有特定权限的实体。它类似于 IAM 用户，但未与特定人员关联。您可以通过切换角色，在 AWS 管理控制台中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义 URL 以代入角色。有关使用角色方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用。

- 临时 IAM 用户权限 – IAM 用户可代入 IAM 角色，暂时获得针对特定任务的不同权限。
- 联合身份用户访问 – 您也可以不创建 IAM 用户，而是使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供商的现有身份。这些用户被称为联合身份用户。在通过身份提供商请求访问权限时，AWS 将为联合身份用户分配角色。有关联合身份用户的更多信息，请参阅 IAM 用户指南 中的联合身份用户和角色。

- 跨账户访问 – 您可以使用 IAM 角色允许其他账户中的某个人（可信任委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南 中的 IAM 角色与基于资源的策略有何不同。
- AWS 服务访问 –服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。在设置一些 AWS 服务环境时，您必须为服务定义要代入的角色。这个服务角色必须包含该服务访问所需的 AWS 资源会用到的所有必要权限。服务角色因服务而异，但只要您满足服务记录在案的要求，许多服务都允许您选择权限。服务角色只在您的账户内提供访问权限，不能用于为访问其他账户中的服务授权。您可以在 IAM 中创建、修改和删除服务角色。例如，您可以创建一个角色，此角色允许 Amazon Redshift 代表您访问 Amazon S3 存储桶，然后将该存储桶中的数据加载到 Amazon Redshift 集群中。有关更多信息，请参阅 IAM 用户指南 中的创建角色以向 AWS 服务委派权限。
- 在 Amazon EC2 上运行的应用程序–对于在 EC2 实例上运行、并发出 AWS CLI 或 AWS API 请求的应用程序，您可以使用 IAM 角色管理它们的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是否使用 IAM 角色，请参阅 IAM 用户指南 中的何时创建 IAM 角色（而不是用户）。

# 使用策略管理访问

您将创建策略并将其附加到 IAM 身份或 AWS 资源，以便控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在某个实体（根用户、IAM 用户或 IAM 角色）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概述。

IAM 管理员可以使用策略来指定哪些用户有权访问 AWS 资源，以及他们可以对这些资源执行哪些操作。每个 IAM 实体（用户或角色）在一开始都没有权限。换言之，默认情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM 策略定义操作的权限，无论您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS 管理控制台、AWS CLI 或 AWS API 获取角色信息。

## 基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、角色或组）的 JSON 权限策略文档。这些策略控制身份可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅 IAM 用户指南 中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是可以附加到 AWS 账户中的多个用户、组和角色的独立策略。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或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 (如 Amazon S3 存储桶) 的 JSON 策略文档。服务管理员可以使用这些策略来定义指定的委托人（账户成员、用户或角色）可以对该资源以及在什么条件执行哪些操作。基于资源的策略是内联策略。没有基于托管资源的策略。

## 访问控制列表 (ACL)

访问控制策略 (ACL) 控制哪些委托人（账户成员、用户或角色）有权访问资源。ACL 类似于基于资源的策略，但它们是唯一不使用 JSON 策略文档格式的策略类型。Amazon S3、AWS WAF 和 Amazon VPC 是

支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅Amazon Simple Storage Service 开发人员指南中的访问控制列表 (ACL) 概述。

## 其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一项高级功能，借助该功能，您可以设置基于身份的策略可以授予 IAM 实体的最大权限（IAM 用户或角色）。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南 中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 是一项服务，用于分组和集中管理您的企业拥有的多个 AWS 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 组织 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南 中的 SCP 工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南 中的会话策略。

## 多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多个策略类型时是否允许请求，请参阅 IAM 用户指南 中的策略评估逻辑。

有关 AWS Batch 的 Identity and Access Management 的更多信息，请转到以下页面。

- AWS Batch 如何与 IAM 协同工作 (p. 105)
- 排查 AWS Batch Identity and Access 的问题 (p. 109)

# AWS Batch 如何与 IAM 协同工作

使用 IAM 来管理 AWS Batch 的访问权限之前，您应该了解哪些 IAM 功能可与 AWS Batch 协同工作。要概括了解 AWS Batch 及其他 AWS 服务如何与 IAM 协同工作，请参阅 IAM 用户指南 中的可与 IAM 协同工作的 AWS 服务。

主题

- AWS Batch 基于身份的策略 (p. 105)
- AWS Batch IAM 角色 (p. 107)

## AWS Batch 基于身份的策略

使用 IAM 基于身份的策略，您可以指定允许或拒绝操作和资源，以及指定在什么条件下允许或拒绝操作。AWS Batch 支持特定操作、资源和条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 IAM JSON 策略元素参考。

### 操作

基于 IAM 身份的策略的 `Action` 元素描述该策略将允许或拒绝的特定操作。策略操作通常与关联的 AWS API 操作同名。此策略用于策略中以授予执行关联操作的权限。

AWS Batch 中的策略操作在操作前使用以下前缀：`batch:`。例如，要授予某人使用 AWS Batch `SubmitJob` API 操作提交 AWS Batch作业的权限，您应将 `batch:SubmitJob` 操作纳入其策略。策略语句

必须包括 Action 或 NotAction 元素。AWS Batch 定义了自己的一组操作，描述了您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示。

```
"Action": [
      "batch:action1",
      "batchaction2"
```

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，请包括以下操作。

```
"Action": "batch:Describe*"
```

要查看 AWS Batch 操作的列表，请参阅 IAM 用户指南 中的 Actions Defined by AWS Batch。

## 资源

Resource 元素指定要向其应用操作的对象。语句必须包含 Resource 或 NotResource 元素。您可使用 ARN 来指定资源，或使用通配符 (*) 以指明该语句适用于所有资源。

AWS Batch 作业定义资源具有以下 ARN。

```
arn:${Partition}:batch:${Region}:${Account}:job-definition/${JobDefinition}
```

有关 ARN 格式的更多信息，请参阅 Amazon 资源名称 (ARN) 和 AWS 服务命名空间。

例如，要在语句中指定 JobDefinitionAlpha 作业定义，请使用以下 ARN：

```
"Resource": "arn:aws:batch:us-east-1:123456789012:job-definition/JobDefinitionAlpha"
```

要指定属于特定账户的所有实例，请使用通配符 (*)。

```
"Resource": "arn:aws:batch:us-east-1:123456789012:job-definition/*"
```

某些 AWS Batch 操作（例如用于创建资源的那些操作）不能在特定资源上执行。在这些情况下，您必须使用通配符 (*)。

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

一个 AWS Batch API 操作涉及多个资源。SubmitJob 将作业提交到作业队列，因此用 IAM 户必须具有使用作业定义和作业队列的权限。要在单个语句中指定多个资源，请使用逗号分隔其 ARN。

```
"Resource": [
      "resource1",
      "resource2"
```

要查看 AWS Batch 资源类型及其 ARN 的列表，请参阅 IAM 用户指南 中的 Resources Defined by AWS Batch。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Actions Defined by AWS Batch。

## 条件键

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以构建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 AWS 使用逻辑 `AND` 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 `OR` 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，仅当 IAM 用户使用其 IAM 用户名进行标记时，您才可为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM 策略元素：变量和标签。

AWS Batch 定义了自己的一组条件键，还支持使用一些全局条件键。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南 中的 AWS 全局条件上下文键。

AWS BatchRegisterJobDefinition 操作支持 `batch:User`、`batch:Privileged` 和 `batch:Image` 条件键。有关更多信息，请参阅 the section called “Supported Resource-Level Permissions” (p. 77)。

要查看 AWS Batch 条件键的列表，请参阅 IAM 用户指南 中的 Condition Keys for AWS Batch。要了解您可以对哪些操作和资源使用条件键，请参阅 Actions Defined by AWS Batch。

### 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 (p. 107)。

AWS Batch 不支持基于资源的策略。

AWS Batch 不支持标记资源或基于标签的访问控制。

## AWS Batch IAM 角色

IAM 角色是 AWS 账户中具有特定权限的实体。

### 使用 AWS Batch 的临时凭证

您可以使用临时凭证进行联合身份登录，代入 IAM 角色或代入跨账户角色。您可以通过调用 AWS STS API 操作（如 AssumeRole 或 GetFederationToken）获得临时安全凭证。

AWS Batch 支持使用临时凭证。

### 服务相关角色

服务相关角色允许 AWS 服务访问其他服务中的资源以代表您完成操作。服务相关角色显示在您的 IAM 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

AWS Batch 不支持服务相关角色。

### 服务角色

此功能允许服务代表您代入服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在您的 IAM 账户中，并由该账户拥有。这意味着 IAM 管理员可以更改此角色的权限。但是，这样操作可能会中断服务的功能。

AWS Batch 支持服务角色。

# AWS Batch 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 AWS Batch 资源的权限。它们还无法使用 AWS 管理控制台、AWS CLI 或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，为用户和角色授予权限，以便对他们所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南 中的 JSON 选项卡上的创建策略。

主题



## 策略最佳实践

基于身份的策略非常强大。它们确定某个人是否可以创建、访问或删除您账户中的 AWS Batch 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议。

- 开始使用 AWS 托管策略 – 要快速开始使用 AWS Batch，请使用 AWS 托管策略，为您的员工授予他们所需的权限。这些策略已在您的账户中提供，并由 AWS 维护和更新。有关更多信息，请参阅 IAM 用户指南 中的利用 AWS 托管策略开始使用权限。
- 授予最低权限 – 创建自定义策略时，仅授予执行任务所需的许可。最开始只授予最低权限，然后根据需要授予其他权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。有关更多信息，请参阅 IAM 用户指南 中的授予最小权限。
- 为敏感操作启用 MFA – 为增强安全性，要求 IAM 用户使用多重身份验证 (MFA) 来访问敏感资源或 API 操作。有关更多信息，请参阅 IAM 用户指南 中的在 AWS 中使用多重身份验证 (MFA)。
- 使用策略条件来增强安全性 – 在切实可行的范围内，定义基于身份的策略在哪些情况下允许访问资源。例如，您可编写条件来指定请求必须来自允许的 IP 地址范围。您也可以编写条件，以便仅允许指定日期或时间范围内的请求，或者要求使用 SSL 或 MFA。有关更多信息，请参阅 IAM 用户指南 中的 IAM JSON 策略元素：Condition。

## 允许用户查看他们自己的权限

此示例显示您可以如何创建策略，以便允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "ViewOwnUserInfo",
               "Effect": "Allow",
               "Action": [
                   "iam:GetUserPolicy",
                   "iam:ListGroupsForUser",
                   "iam:ListAttachedUserPolicies",
                   "iam:ListUserPolicies",
                   "iam:GetUser"
               ],
               "Resource": [
                   "arn:aws:iam::*:user/${aws:username}"
               ]
           },
           {
               "Sid": "NavigateInConsole",
               "Effect": "Allow",
               "Action": [
                   "iam:GetGroupPolicy",
                   "iam:GetPolicyVersion",
                   "iam:GetPolicy",
                   "iam:ListAttachedGroupPolicies",
                   "iam:ListGroupPolicies",
                   "iam:ListPolicyVersions",
                   "iam:ListPolicies",
                   "iam:ListUsers"
               ],
```

```
            "Resource": "*"
        }
    ]
}
```

# 排查 AWS Batch Identity and Access 的问题

使用以下信息可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

如果 AWS 管理控制台 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

当 `mateojackson` IAM 用户尝试使用控制台查看有关计算环境的详细信息，但不具有 `batch:DescribeComputeEnvironments` 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:DescribeComputeEnvironments on resource: my-example-compute-environment
```

在这种情况下，Mateo 请求管理员更新其策略，以允许他使用 `batch:DescribeComputeEnvironments` 操作访问 `my-example-compute-environment` 资源。

## 我无权执行 iam:PassRole

如果您收到错误消息，提示您无权执行 `iam:PassRole` 操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。请求那个人更新您的策略，以便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到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 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，Mary 请求她的管理员来更新其策略，以允许她执行 `iam:PassRole` 操作。

## 我想要查看我的访问密钥

创建 IAM 用户访问密钥之后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

Important

请不要向第三方提供访问密钥，甚至为了帮助找到您的规范用户 ID 也不能提供。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果您丢失了秘密访问密钥，则必须向您的 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南 中的管理访问密钥。

## 我是管理员并希望允许其他人访问 AWS Batch

要允许其他人访问 AWS Batch，您必须为需要访问权限的人员或应用程序创建 IAM 实体（用户或角色）。他们（它们）将使用该实体的凭证访问 AWS。然后，您必须将策略附加到实体，以便在 AWS Batch 中为他们（它们）授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南 中的创建您的第一个 IAM 委托用户和组。

## 我想要允许我的 AWS 账户之外的用户访问我的 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 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容。

- 要了解 AWS Batch 是否支持这些功能，请参阅AWS Batch 如何与 IAM 协同工作 (p. 105)。
- 要了解如何向您拥有的 AWS 账户中的资源提供访问权限，请参阅 IAM 用户指南 中的对您拥有的 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何向第三方 AWS 账户提供对您的资源的访问权限，请参阅 IAM 用户指南 中的向第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南 中的向经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南 中的 IAM 角色和基于资源的策略有何不同。

# AWS Batch 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审计员将评估 AWS Batch 的安全性和合规性。其中包括 SOC、PCI、ISO、HIPAA BAA、MTCS 等。

有关特定合规性计划范围内的 AWS 服务列表，请参阅合规性计划范围内的 AWS 服务。有关常规信息，请参阅 AWS 合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅下载 AWS 构件中的报告。

您在使用 AWS Batch 时的合规性责任由您数据的敏感性、您公司的合规性目标以及适用的法律法规决定。AWS 提供以下资源来帮助满足合规性。

- 安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。

- AWS Config – 此 AWS 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – 此 AWS 服务提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

# AWS Batch 中的基础设施安全性

作为一项托管服务，AWS Batch 由 Amazon Web Services：安全流程概述白皮书中所述的 AWS 全球网络安全程序提供保护。

您可以使用 AWS 发布的 API 调用通过网络访问 AWS Batch。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

# AWS Batch 服务限制

下表提供了无法更改的 AWS Batch 服务限制。每个限制都特定于具体区域。

| 资源 | 限制 |
| --- | --- |
| 最大作业队列数。有关更多信息，请参阅 作业队列 (p. 48)。 | 20 |
| 最大计算环境数。有关更多信息，请参阅 计算环境 (p. 52)。 | 50 |
| 每个作业队列的最大计算环境数 | 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 |

根据您使用 AWS Batch 的方式，可能会适用额外的配额。要了解 Amazon EC2 配额的信息，请参阅 AWS General Reference 中的 Amazon EC2 服务配额。要了解 Amazon ECS 配额的信息，请参阅 AWS General Reference 中的 Amazon ECS 服务配额。

# 排查 AWS Batch 问题

您可能会发现需要排查计算环境、作业队列、作业定义或作业的问题。本章帮助您排查和修复您的 AWS Batch 环境的问题。

## INVALID 计算环境

可能会错误地配置托管计算环境，使其进入 INVALID 状态，而无法接受作业进行放置。以下各节介绍了可能的原因以及修复方法。

### 角色名称或 ARN 不正确

无效计算环境的最常见原因是 AWS Batch 服务角色或 Amazon EC2 Spot 队组角色的名称或 ARN 不正确。对于使用 AWS CLI 或 AWS 开发工具包创建的计算环境，这更是一个问题；当您在 AWS 管理控制台中创建计算环境时，AWS Batch 可以帮助您选择正确的服务或 Spot 队组角色，您不会拼错名称或使用格式错误的 ARN。

但是，如果您在 AWS CLI 命令或软件开发工具包代码中手动键入 IAM 的名称或 ARN，则 AWS Batch 将无法验证字符串，从而接受该错误值并尝试创建环境。创建环境失败后，环境将进入 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 Spot 队列角色：

```
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
```

导致此问题的一个常见原因是您在使用 AWS CLI 或 AWS 开发工具包时仅指定 IAM 角色的名称，而不是完整 ARN。这是因为根据您创建角色的方式，ARN 可能包含 service-role 路径前缀。例如，如果您使用 AWS Batch Service IAM Role (p. 83)中的步骤手动创建 AWS Batch 服务角色，您的服务角色 ARN 将如下所示：

```
arn:aws:iam::123456789012:role/AWSBatchServiceRole
```

但是，如果您在控制台首次运行向导的过程中创建了服务角色，那么您的服务角色 ARN 将如下所示：

```
arn:aws:iam::123456789012:role/service-role/AWSBatchServiceRole
```

如果您在使用 AWS CLI 或 AWS 开发工具包时仅指定 IAM 角色的名称，AWS Batch 将假定您的 ARN 不使用 service-role 路径前缀。因此，我们建议您在创建计算环境时为 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 (p. 114)。

# 修复 INVALID 计算环境

当您拥有处于 INVALID 状态的计算环境时，您应该更新它以修复无效参数。对于 角色名称或 ARN 不正确 (p. 113) 的情况，您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. 从 https://console.aws.amazon.com/batch/ 打开 AWS Batch 控制台。
2. 从导航栏中，选择要使用的区域。
3. 在导航窗格中，选择 Compute environments。
4. 在 Compute environments 页面上，选择要编辑的计算环境旁边的单选按钮，然后选择 Edit。
5. 在更新计算环境页面上，对于服务角色，请选择要用于您的计算环境的 IAM 角色。AWS Batch 控制台仅显示与计算环境具有正确信任关系的角色。
6. 选择 Save 以更新您的计算环境。

# 作业在 RUNNABLE 状态卡住

如果您的计算环境包含计算资源，但您的作业在 RUNNABLE 状态停止，无法继续，则表明出现问题，导致作业无法被实际放置在计算资源上。以下是此问题的一些常见原因：

您的计算资源上未配置 awslogs 日志驱动程序

AWS Batch 任务将它们的日志信息发送到 CloudWatch Logs。为了做到这一点，您必须将您的计算资源配置为使用 awslogs 日志驱动程序。如果您的计算资源 AMI 基于经 Amazon ECS 优化的 AMI（或 Amazon Linux），则该驱动程序默认向 ecs-init 程序包注册。如果使用不同的基础 AMI，则必须确保在启动 Amazon ECS 容器代理时，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 (p. 53) 和 创建计算资源 AMI (p. 54)。

资源不足

如果您的作业定义指定的 CPU 或内存资源超出可分配的计算资源量，那么您的作业将永远不会被放置。例如，如果您的作业指定了 4 GiB 的内存，而您的可用计算资源少于此数量，则无法将作业放置在这些计算资源上。在这种情况下，您必须减少作业定义中指定的内存，或者向您的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i一些内存。有关更多信息，请参阅 计算资源内存管理 (p. 71)。

没有 Internet 访问权限，无法获得计算资源

计算资源需要访问权限以便与 Amazon ECS 服务终端节点通信。这可以通过接口 VPC 终端节点或通过具有公有 IP 地址的计算资源完成。

有关接口 VPC 终端节点的更多信息，请参阅 Amazon Elastic Container Service Developer Guide 中的 Amazon ECS 接口 VPC 终端节点 (AWS PrivateLink)。

如果您没有配置接口 VPC 终端节点并且您的计算资源没有公有 IP 地址，则它们必须使用网络地址转换 (NAT) 来提供此访问。有关更多信息，请参阅 Amazon VPC 用户指南 中的 NAT 网关。有关更多信息，请参阅教程：为您的计算环境创建带有公有和私有子网的 VPC (p. 100)。

已达到 Amazon EC2 实例限制

您的账户可在 AWS 区域中启动的 Amazon EC2 实例的数量由您的 EC2 实例限制决定。某些实例类型也有基于实例类型的限制。有关您的账户的 Amazon EC2 实例限制的更多信息（包括如何请求限制提升），请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的 Amazon EC2 服务限制。

有关诊断处于 RUNNABLE 状态的作业的详细信息，请参阅 AWS 知识中心中的我的 AWS Batch 作业为什么卡在 RUNNABLE 状态？。

# 在创建时未标记的 Spot 实例

从 2017 年 10 月 25 日起支持对 AWS Batch 计算资源标记 Spot 实例。在提供此支持之前，建议对 Amazon EC2 Spot 队组角色使用的 IAM 托管策略 (`AmazonEC2SpotFleetRole`) 不包含在 Spot 实例启动时对其进行标记的权限。推荐使用的新 IAM 托管策略称为 `AmazonEC2SpotFleetTaggingRole`。

要解决创建时为 Spot 实例添加标签的问题，请按以下过程操作，将当前推荐的 IAM 托管策略应用于您的 Amazon EC2 Spot 队组角色，这样，将来使用该角色创建的任何 Spot 实例都将有权在创建时应用实例标签。

### 将当前 IAM 托管策略应用于您的 Amazon EC2 Spot 队组角色

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择角色，然后选择您的 Amazon EC2 Spot 队组角色。
3. 选择 Attach policy。
4. 选择 AmazonEC2SpotFleetTaggingRole，然后选择 Attach policy。
5. 再次选择您的 Amazon EC2 Spot 队列角色，以移除以前的策略。
6. 选择 AmazonEC2SpotFleetRole 策略右侧的 x，然后选择 Detach。

# 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您发送给我们的反馈意见。

| update-history-change | update-history-description | update-history-date |
| --- | --- | --- |
| 分配策略 | AWS Batch 增加了对多个用于选择实例类型的策略的支持。 | October 16, 2019 |
| EFA 支持 | AWS Batch 增加了对 Elastic Fabric Adapter (EFA) 设备的支持。 | August 2, 2019 |
| GPU 计划 | 您可以使用 GPU 作业指定每个作业所需的 GPU 数量。 AWS Batch 将扩展适合您的作业的实例。 | April 4, 2019 |
| 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。 | November 19, 2018 |
| 资源级权限 | AWS Batch 现在支持多个 API 操作的资源级权限。 | November 12, 2018 |
| Amazon EC2 启动模板支持 | AWS Batch 增加了对在计算环境中使用启动模板的支持。 | November 12, 2018 |
| AWS Batch 任务超时时间 | 您可以为任务配置超时时间，以便在某个任务运行的时间超过超时时间时让 AWS Batch 终止该任务。 | April 5, 2018 |
| 作为 CloudWatch Events 目标的 AWS Batch 任务 | AWS Batch 任务可作为 CloudWatch Events 目标提供。通过使用可快速设置的简单规则，您可以匹配事件并提交 AWS Batch 任务以响应这些事件。 | March 1, 2018 |
| 适用于 AWS Batch 的 CloudTrail 审核 | CloudTrail 可以审核对 AWS Batch API 操作的调用。 | January 10, 2018 |
| 数组作业 | AWS Batch 支持数组作业，该作业对于参数清除和蒙特卡洛工作负载非常有用。 | November 28, 2017 |
| 扩展了 AWS Batch 标记 | AWS Batch 使您能够针对托管计算环境中启动的 Amazon EC2 Spot 实例指定标签。 | October 26, 2017 |
| CloudWatch Events 的 AWS Batch 事件流 | 使用 CloudWatch Events 的 AWS Batch 事件流近乎实时地接收通知，了解已提交到任务队列的任务的当前状态。 | October 24, 2017 |

| | | |
|---|---|---|
| 自动作业重试 | 您可以将重试策略应用于作业和作业定义，这将使作业能够在失败时自动重试。 | March 28, 2017 |
| AWS Batch 正式发布 (p. 116) | 利用 AWS Batch，您可以在 AWS 云上运行批量计算工作负载。 | January 5, 2017 |

如果我们为英文版本指南提供翻译，那么如果存在任何冲突，将以英文版本指南为准。在提供翻译时使用机器翻译。