EC2 映像生成器

用户指南

EC2 映像生成器: 用户指南

Table of Contents

什么是 EC2 映像生成器？

EC2 映像生成器 是一项完全托管的 AWS 服务，它可以轻松自动创建、管理和部署自定义、安全且最新的“黄金”服务器映像，这些映像是预先安装的，并预配置了软件和设置以符合特定的 IT 标准。

您可以使用 AWS 管理控制台、AWS CLI 或 API 在您的 AWS 账户中创建“黄金”映像。在使用 AWS 管理控制台时，镜像生成器 向导将指导您完成以下步骤：

- 提供起始构件
- 添加和删除软件
- 自定义设置和脚本
- 运行选定的测试
- 将映像分发到 AWS 区域

您生成的映像是在您的账户中创建的，您可以将其配置为定期安装操作系统补丁。

要对映像部署进行故障排除和调试，您可以配置生成日志以添加到 Amazon Simple Storage Service (Amazon S3) 存储桶中。您还可以将实例生成应用程序配置为将日志发送到 CloudWatch。为了接收映像生成状态通知，并将 Amazon Elastic Compute Cloud (Amazon EC2) 密钥对与实例关联以执行手动调试和检查，您可以配置 SNS 主题。

除了最终映像以外，镜像生成器 还会创建映像配方，这是生成的源映像和组件组合。您可以将映像配方与现有的源代码版本控制系统和持续集成/持续部署管道一起使用，以实施可重复的自动化。

本节内容

EC2 映像生成器 功能

EC2 映像生成器 提供以下功能：

提高生产效率，并减少生成符合要求的最新映像的操作

通过自动完成生成管道，镜像生成器 减少了大规模创建和管理映像所需的工作量。您可以提供生成执行计划首选项以自动完成生成。自动化降低了使用最新操作系统补丁维护软件的运营成本。

增加服务正常运行时间

镜像生成器 允许您在部署之前使用 AWS 提供的测试和自定义的测试对映像进行测试。只有在成功完成所有配置的测试时，AWS 才会分发映像。

提高部署安全性

通过使用 镜像生成器，您可以创建映像以消除不必要的组件安全漏洞风险。您可以应用 AWS 安全设置，以创建符合行业和内部安全标准的安全的现成映像。镜像生成器 还为受管制行业的公司提供了一组设置。您可

以使用这些设置，以帮助您快速轻松地生成符合 STIG 标准的映像。有关通过 镜像生成器 提供的 STIG 组件的完整列表，请参阅 EC2 映像生成器 STIG 组件 (p. 47)。

集中的实施和跟踪管理

通过使用与 AWS Organizations 的内置集成，您可以通过 镜像生成器 实施策略以仅限账户从批准的 AMI 中运行实例。

支持的操作系统

镜像生成器 支持以下操作系统：

- Amazon Linux 2
- Windows Server 2019/2016/2012 R2
- Windows Server 版本 1909
- Red Hat Enterprise Linux (RHEL) 8 和 7
- CentOS 8 和 7（CentOS 8 未在 AWS Marketplace 中作为公有 AMI 提供。您可以使用 VMIE 自带您的 CentOS 8 AMI）
- Ubuntu 18 和 16
- SUSE Linux Enterprise Server (SUSE) 15

支持的映像格式

您可以选择现有的 AMI 以作为生成映像的起点。

概念

以下术语和概念对您了解和使用 EC2 映像生成器 非常有用。

AMI

Amazon 系统映像 (AMI) 是 Amazon EC2 中的基本部署单元。AMI 是一个预配置的虚拟机映像，其中包含用于部署 EC2 实例的操作系统和预装软件。有关更多信息，请参阅 Amazon 系统映像 (AMI)。

映像管道

映像管道是用于在 AWS 上生成安全操作系统映像的自动化配置。镜像生成器 映像管道与映像配方相关联，该配方定义映像生成生命周期的生成、验证和测试阶段。映像管道可以与定义映像生成位置的基础设施配置相关联。您可以定义一些属性，例如，实例类型、子网、安全组、日志记录以及其他基础设施相关配置。您也可以将映像管道与分发配置相关联，以定义所需的映像部署方式。

映像配方

镜像生成器 映像配方是一个文档，用于定义源映像以及要应用于源映像的组件以生成输出映像所需的配置。您可以使用映像配方复制生成。可以使用控制台向导、AWS CLI 或 API 共享和编辑 镜像生成器 映像配方以及创建分支。您可以将映像配方与版本控制软件一起使用，以维护可共享且进行版本控制的映像配方。

源映像

源映像是映像配方文档中与组件一起使用的选定映像和操作系统。源映像和组件组合在一起，定义了生成输出映像所需的配置。

组件

组件定义在创建映像之前改变实例（生成组件）或测试从创建的映像启动的实例（测试组件）所需的步骤及顺序。组件是一个声明性的纯文本 YAML 文档，描述了要执行的一系列步骤。组件通过组件管理应用程序在实例上执行。组件管理应用程序会解析文档并执行所需的步骤。创建组件后，需要根据映像配方将它们分成一个或多个一组，用于定义生成和测试虚拟机映像的执行计划。您可以使用 AWS 拥有和管理的公共组件，也可以创建自己的组件。有关组件的更多详细信息，请参阅 EC2 映像生成器 组件管理器 (p. 32)。

文档

这是一种声明性文档，它使用 YAML 格式列出在实例上生成、验证和测试 AMI 的执行步骤。该文档是配置管理应用程序的输入，该应用程序在 Amazon EC2 实例本地运行以执行文档步骤。

执行阶段

EC2 映像生成器 有两个执行阶段：生成和测试。组件有三个可能的阶段：生成、验证和测试。映像管道启动时，镜像生成器 处于生成阶段。在此阶段，会启动一个实例，并将映像配方中定义的所有组件下载到该实例。组件按照映像配方指定的顺序执行。对于每个组件，组件管理应用程序会解析相关文档并执行生成和验证阶段。如果成功，则会关闭实例并创建映像，而 镜像生成器 将继续执行测试阶段。在测试阶段，将从先前创建的映像启动实例，并将组件下载到该实例。组件管理应用程序将解析文档并执行测试阶段。镜像生成器 中的生成阶段对应于组件中的生成和验证阶段，而测试阶段对应于组件中的测试阶段。

定价

使用 EC2 映像生成器 不会产生任何费用。启动 Amazon EC2 实例以及在 Amazon S3 上存储日志、使用 Amazon Inspector 验证映像以及 Amazon EBS 快照的 AMI 存储可能会产生相应的费用，具体取决于您的映像的配置。如果启用 Systems Manager 高级套餐并运行具有本地激活的 EC2 实例，则可能会通过 Systems Manager 向您收取资源费用。

相关 AWS 服务

EC2 映像生成器 使用其他 AWS 服务生成映像。根据 镜像生成器 映像配方配置，可能使用以下服务。

AWS License Manager

AWS License Manager 允许您从账户许可证配置存储中创建和应用许可证配置。对于每个 AMI，您可以使用 镜像生成器 附加您的 AWS 账户有权访问的预先存在的许可证配置，以作为 镜像生成器 工作流程的一部分。只能将许可证配置应用于 AMI。镜像生成器 只能使用预先存在的许可证配置，而不能直接创建或修改许可证配置。不会在您的账户中必须启用的 AWS 区域之间复制 License Manager 设置，例如，在 `ap-east-1` (HKG) 和 `me-south-1` (BAH) 区域之间。

AWS Systems Manager (SSM) Automation

Systems Manager Automation 文档定义 Systems Manager 对托管实例和 AWS 资源执行的操作。SSM 文档使用 JSON 或 YAML，并包含您指定的步骤和参数。您指定的步骤按顺序运行。Automation 文档是 Automation 类型的 Systems Manager 文档，与 Command 和 Policy 文档相对。有关更多信息，请参阅 AWS Systems Manager Automation。

Amazon CloudWatch Logs

您可以使用 Amazon CloudWatch Logs 来监控、存储和访问来自 Amazon EC2 实例、AWS CloudTrail、Amazon Route 53 和其他来源的日志文件。

EC2 映像生成器 的工作原理

在使用 EC2 映像生成器 控制台创建黄金映像时，向导将指导您完成以下步骤。

1. 选择源映像。 您选择一个源操作系统映像，例如，现有的 AMI。
2. 创建映像配方。 您添加组件以创建映像管道的映像配方。组件是映像配方使用的构建块，例如，安装软件包、安全强化步骤和测试。选定的操作系统和组件组成了映像配方。组件是按指定顺序安装的，在选择后，无法重新排序。
3. 输出。镜像生成器 使用选定的输出格式创建操作系统映像。
4. 分发。 在映像管道中的映像通过测试后，您可以将其分发到选定的 AWS 区域。

您从黄金映像生成的映像位于您的 AWS 账户中。您可以输入生成计划，以便配置映像管道以生成更新和修补的 AMI 版本。在生成完成后，您可以通过 Amazon Simple Notification Service (SNS) 接收通知。除了生成最终映像以外，镜像生成器 还会生成一个映像配方，可以将其与现有版本控制系统和持续集成/持续部署 (CI/CD) 管道一起使用以实现可重复的自动化。您可以共享映像配方和创建新的映像配方版本。

本节内容

AMI 组件

Amazon 系统映像 (AMI) 是 Amazon EC2 中的基本部署单元。它是预配置的虚拟机 (VM) 映像，其中包含用于部署 EC2 实例的操作系统和软件。

AMI 包括以下组件：

- 虚拟机根卷的模板。在启动 EC2 虚拟机时，根设备卷包含用于引导实例的映像。在使用实例存储时，根设备是通过 Amazon S3 中的模板创建的实例存储卷。有关更多信息，请参阅 Amazon EC2 根设备卷。
- 在使用 Amazon EBS 时，根设备是通过 EBS 快照创建的 EBS 卷。
- 启动权限，用于确定可以使用 AMI 启动虚拟机的 AWS 账户。
- 块储存设备映射数据，用于指定在启动后附加到实例的卷。
- 每个账户的每个区域的唯一资源标识符。
- 元数据负载（例如标签）和属性（例如区域、操作系统、架构、根设备类型、提供程序、启动权限、根设备存储以及签名状态）。
- AMI 签名，用于防止未经授权的篡改。有关更多信息，请参阅实例身份文档。

默认配额

要查看 EC2 映像生成器 的默认配额，请参阅 EC2 Image Builder 终端节点和配额。

AWS 区域和终端节点

要查看 EC2 映像生成器 的服务终端节点，请参阅 EC2 Image Builder 终端节点和配额。

日志

EC2 映像生成器 与 AWS 服务集成可提供监控功能，帮助您排查映像生成过程中的问题。镜像生成器 可以跟踪显示映像生成过程中每个步骤的进度。您可以将映像生成应用程序配置为将日志发送到 CloudWatch 以及您提供的 S3 位置。有关 CloudWatch Logs 的更多信息，请参阅什么是 Amazon CloudWatch Logs？。

默认情况下，系统会启用 CloudWatch 日志记录支持。日志会保留在实例上并流式传输到 CloudWatch。在创建 AMI 的过程中，系统会从实例中删除日志。这些日志会流式传输到以下 LogStream：

- LogGroup："/aws/imagebuilder/**<ImageName>**
- LogStream：**<ImageVersion>**/**<ImageBuildVersion>**["x.x.x/x"]

可以通过删除与实例配置文件关联的以下权限，选择不执行 CloudWatch 流式传输。

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:CreateLogGroup",
            "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"
    }
]
```

对于高级故障排除，您可以使用 AWS Systems Manager (SSM) Run Command 运行预定义的命令和脚本。有关更多信息，请参阅 EC2 映像生成器 故障排除 (p. 73)。

组件管理器

镜像生成器 使用组件管理应用程序，以帮助您编排复杂的工作流程，修改系统配置以及测试系统，而无需编写代码。该应用程序使用声明性的文档架构。由于它是一个单独的应用程序，因此，它不需要进行额外的服务器设置。它可以在任何云基础设施和本地运行。

EC2 映像生成器 使用该应用程序执行所有实例上的活动，例如生成、验证和测试。您定义一个文档以描述如何生成、验证和测试映像。EC2 映像生成器 将组件发送到实例，该应用程序执行定义的阶段、步骤和操作以解释组件并将其应用于实例。在完成后，该应用程序向 EC2 映像生成器 发送摘要。如果在管道配置中指定了 S3 存储桶，它还会将详细的执行输出发送到 Amazon S3。然后，EC2 映像生成器 使用 AWS 强化和清理映像的最佳实践。

- 生成阶段。修改映像。例如，您可以配置映像以安装应用程序或修改操作系统防火墙设置。在创建映像之前，将执行验证阶段以作为生成阶段的一部分。
- 测试阶段。在创建新的映像后，将对该映像执行测试。

EC2 映像生成器 按如下方式使用组件管理应用程序。

1. 您定义一个 EC2 映像生成器 组件，它是一个描述如何生成、验证和测试映像的文档。

2. EC2 映像生成器 将文档和该应用程序复制到实例以分派要执行的工作。
3. 该应用程序执行文档中定义的阶段、步骤和操作。

有关 镜像生成器 在编排工作流程时使用的组件管理器的更多信息（包括有关文档、支持的操作模块和 STIG 的信息），请参阅 EC2 映像生成器 组件管理器 (p. 32)。

创建的资源

在创建管道时，不会创建 镜像生成器 外部的资源。只有在通过管道计划、镜像生成器 控制台中的 Run Pipeline (运行管道) 操作、`StartImagePipelineExecution` API 或 `CreateImage` API 创建映像时，才会创建 镜像生成器 外部的资源。在映像创建期间，将创建以下资源。

- Amazon EC2 实例
- SSM Inventory 关联（通过 SSM State Manager）（如果启用了 `EnhancedImageMetadata`）
- Amazon EC2 AMI
- EBS 快照（与 Amazon EC2 AMI 关联）

在创建 AMI 后，将删除 Amazon EBS 快照和 Amazon EC2 AMI 以外的所有资源。

EC2 映像生成器 入门

本节包含设置您的环境以及使用 EC2 映像生成器 创建映像管道所需的信息。

本节内容

先决条件

必须验证以下先决条件，才能使用 EC2 映像生成器 创建映像管道。

EC2 映像生成器 服务相关角色

EC2 映像生成器 使用服务相关角色代表您为其他 AWS 服务授予权限。您无需手动创建服务相关角色。在 AWS 管理控制台、AWS CLI 或 AWS API 中创建第一个 镜像生成器 资源时，镜像生成器 将为您创建服务相关角色。有关 镜像生成器 在您的账户中创建的服务相关角色的更多信息，请参阅将服务相关角色用于 EC2 映像生成器 (p. 64)。

配置要求

- EC2 映像生成器 不支持将加密的 AMI 作为管道的源或输出映像。
- 您必须在基础设施配置中指定一个 VPC。镜像生成器 不支持 EC2-Classic。
- 用于通过 Image Builder 生成映像和运行测试的实例必须有权访问 Systems Manager 服务。所有生成活动都是由 SSM Automation 编排的。如果 SSM 代理尚不存在，将在源映像上安装该代理，并在创建映像之前将其删除。

AWS Identity and Access Management (IAM)

与实例配置文件关联的 IAM 角色必须有权运行映像中包含的生成和测试组件。必须将以下 IAM 角色策略附加到与实例配置文件关联的 IAM 角色：EC2InstanceProfileForImageBuilder 和 AmazonSSMManagedInstanceCore。

如果配置日志记录，在基础设施配置中指定的实例配置文件必须具有目标存储桶 (`arn:aws:s3:::BucketName/*`) 的 `s3:PutObject` 权限。

访问 EC2 映像生成器

您可以通过以下界面之一管理 EC2 映像生成器。

- EC2 映像生成器 控制台登录页面。从 EC2 映像生成器 登录页。
- AWS 命令行界面 (AWS CLI)。您可以使用 AWS CLI 访问 AWS API 操作。有关更多信息，请参阅 AWS Command Line Interface 用户指南 中的安装 AWS 命令行界面。
- AWS 开发工具包工具。您可以使用 AWS 开发工具包和工具通过首选的语言访问和管理 镜像生成器。

使用 EC2 映像生成器 控制台生成操作系统映像并自动完成部署

以下步骤指导您完成从 EC2 映像生成器 控制台中使用 镜像生成器 部署映像的过程。

1. 从 EC2 映像生成器 登录页面中，选择 Create image pipeline (创建映像管道)。
2. 以下选项卡包含有关一些页面的信息，您必须为这些页面提供输入以创建映像管道。

   Define recipe

   a. 在 Define Recipe (定义配方) 页面上，创建一个映像配方，其中包含源映像和组件。

      i. 选择源映像。源映像包含映像操作系统和要配置的映像。在选择映像操作系统后，从以下选项中进行选择以选择要配置的映像。

         A. 从管理的映像中选择一个映像，其中包括帮助您开始使用的 镜像生成器 映像、您已创建的映像以及已与您共享的映像。要选择映像，请在文本框中输入映像 ARN，或选择 Browse images (浏览映像) 以查看管理的映像。AWS 提供的所有管理映像都是 64 位操作系统。

         B. 输入 AMI ID 以使用自定义 AMI。

         如果希望 镜像生成器 使用语义版本控制以设置映像的版本号，请选中“Always build latest version”(始终生成最新的版本) 复选框。如果未选中该框，镜像生成器 将始终使用相同的版本号。选中该框并不会在选定的映像版本具有更新时启动自动生成，除非您已使用 Configure Pipeline (配置管道) 下的作业计划程序将生成管道设置为自动运行。

      ii. 选择 Build components (生成组件)。组件是在生成映像时映像配方使用的安装软件包、安全强化步骤和测试。在创建映像配方后，无法修改或替换其组件。如果要更新映像配方中的组件，请创建新的映像配方或映像配方版本。

         > **Important**
         >
         > 组件是按选择顺序安装的。在选择组件后，您无法对其重新排序。

         组件包括两种组件类型。

         A. 生成组件。生成组件是安装软件包和安全强化步骤。您可以输入组件 ARN，或者浏览并从 镜像生成器 组件列表中选择以帮助您开始使用。要创建新的组件，请选择 Create Component (创建组件)。有关如何创建组件的信息，请参阅创建新的组件 (p. 12)。按照您希望组件在映像生成管道中运行的顺序输入或选择组件。

         B. 测试组件。测试组件是对映像管道生成的输出映像执行的测试。请输入测试组件 ARN，或者浏览并从 镜像生成器 测试组件列表中选择以帮助您开始使用。要创建新的组件，请选择 Create Component (创建组件)。有关如何创建组件的信息，请参阅创建新的组件 (p. 12)。按照您希望组件在映像生成管道中运行的顺序输入或选择组件。

         在输入源映像和组件后，选择 Next (下一步)。

   Configure pipeline

   a. 从 Configure Pipeline (配置管道) 页面中，定义映像管道基础设施和生成计划。

      i. 在 Pipeline details (管道详细信息) 下提供以下规范。

         A. 输入映像管道的 Name (名称)。您必须使用唯一的映像管道名称。

         B. 为映像部署管道提供可选的 Description (描述)。

         C. 选择要与实例配置文件关联的 IAM 角色或 Create a new role (创建新的角色)。如果创建新的角色，镜像生成器 将转到 IAM 控制台。首先，使用以下 IAM 角色策略（您必须将这两个策略都附加上）：EC2InstanceProfileForImageBuilder 和 AmazonSSMManagedInstanceCore。

            > **Important**
            >
            > 确保您的角色有权运行映像中包含的生成和测试组件。如果配置了日志记录，在 InfrastructureConfiguration 中指定的实例配置文件必须具有目标存储桶的所需权限

(s3:PutObject)。要实现该目的，您可以在与实例配置文件关联的角色中包括内联策略，或者将 S3FullAccess 托管策略附加到实例配置文件。

ii. 选择一个运行映像管道的 Build schedule (生成计划)。

A. 如果选择 Manual (手动)，您可以在自己选择的任何时间运行管道。在您希望运行管道时，请在 Pipeline details (管道详细信息) 页面上选择 Run pipeline (运行管道)。

B. 如果选择 Schedule builder (计划生成器)，您可以使用作业计划程序将生成管道设置为自动运行。在 Run pipeline every (运行管道频率) 后面输入频率。您可以选择每天、每周或每月运行一次管道。要将生成管道设置为从最新的映像版本中生成，您必须选中 Define Recipe (定义配方) 下的 Always build latest version (始终生成最新的版本) 复选框。

C. 如果选择 CRON expression (CRON 表达式)，您可以设置生成管道以使用指定运行时间和间隔的语法来运行该管道。在文本框中输入表达式。

要查看在运行生成管道时创建的资源，请参阅创建的资源 (p. 6)。

iii. （可选）输入 Infrastructure (基础设施) 规范以定义映像的基础设施。这些设置与在您的账户中启动以生成映像的 EC2 实例相关联。

A. 选择一种 Instance type (实例类型)。选定的实例类型应满足计划在实例上运行的软件的要求。有关 EC2 实例类型的更多信息，请参阅 EC2 用户指南 中的实例类型。

B. 如果希望从 镜像生成器 接收有关在映像管道中执行的任何步骤的通知和警报，您可以输入用于接收 AWS Simple Notification Service (SNS) 通知的 SNS 主题 ARN。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南。

C. 在 Troubleshooting settings (故障排除设置) 下面，提供以下信息。如果映像生成失败，在对实例执行故障排除时，这些设置是非常有用的。

I. 在 Key pair name (密钥对名称) 下，从下拉列表中选择一个现有的密钥对或创建新的密钥对。

1. 如果选择 Create key pair name (创建密钥对名称) 以创建新的密钥对，您将转至 Amazon EC2 控制台。
2. 从 Amazon EC2 控制台中，选择 Create a new key pair (创建新的密钥对)。
3. 输入密钥对的名称。
4. 然后，选择 Download Key Pair (下载密钥对)。

Important

这是保存私有密钥文件的唯一机会，因此，请务必将其下载并保存到安全的地方。您必须在启动实例时提供密钥对的名称，并在每次连接到实例时提供相应的私有密钥。

5. 返回到 镜像生成器 控制台，然后选择 Key pair name (密钥对名称) 下拉列表旁边的刷新图标。将在下拉列表中显示新创建的密钥对。有关密钥对的更多信息，请参阅 Amazon EC2 密钥对。

II. 选中或取消选中 Terminate your instance upon failure (在失败时终止实例) 复选框，以选择是否要执行该操作。如果您希望能够在映像生成失败时对实例进行故障排除，请确保该复选框未选中。

Note

如果未选择在失败时终止实例的选项，在生成失败时，不会从您的账户中删除用于启动实例的 Auto Scaling 组和启动模板。您必须手动删除 Auto Scaling 组和启动模板资源。

III. 在 S3 Logs (S3 日志) 下，选择要将实例日志文件发送到的 S3 存储桶。要浏览并选择 Amazon S3 存储桶位置，请选择 Browse S3 (浏览 S3)。这些日志显示映像生成过程中 EC2 实例上的活动的步骤和错误消息。

IV. 如果要选择一个 VPC 以启动实例，请在 Advanced Settings (高级设置) 下提供以下信息。

1. 选择要在其中启动实例的 Virtual Private Cloud。有关 VPC 的更多信息，请参阅 VPC 用户指南。您也可以选择 Create a new VPC (创建新的 VPC)。如果选择执行该操作，将会转到 VPC 控制台。要允许在 VPC 和 Internet 之间进行通信，您必须使用 Internet

网关启用该连接。要将 Internet 网关添加到 VPC 中，请按照 Amazon VPC 用户指南的创建并附加互联网网关中的步骤进行操作。

2. 如果选择一个 VPC，请选择与选定的 VPC 关联的 Public subnet ID (公有子网 ID)，或选择 Create new subnet (创建新的子网)。有关更多信息，请参阅 VPC 和子网。
3. 如果选择一个 VPC，请选择与 VPC 关联的 Security groups (安全组)，或选择 Create a new security group (创建新的安全组)。有关安全组的帮助，请参阅 您的 VPC 的安全组。

   Note

   EC2 映像生成器 不支持 EC2-Classic。如果在您的账户使用 EC2-Classic 或没有默认 VPC 的 AWS 区域中生成映像，您必须选择一种 VPC 配置。有关更多信息，请参阅 Amazon VPC 用户指南 中的默认 VPC 和默认子网。

在输入所有基础设施规范后，选择 Next (下一步)。

Configure additional settings

a. 从 Configure additional settings (配置其他设置) 页面中，您可以选择定义测试和分发设置以及在生成映像后使用的其他可选配置参数。如果要定义这些配置，请提供以下信息。
   i. 在 Associate license configuration to AMI (将许可证配置与 AMI 关联) 下，您可以选择将输出 AMI 与使用 AWS License Manager 创建的预先存在的许可证配置相关联。从下拉列表中选择一个或多个唯一的许可证配置 ID。如果要创建新的许可证配置，请选择 Create new License Configuration (创建新的许可证配置)。这会转到 License Manager 控制台。有关更多信息，请参阅什么是 AWS License Manager？。
   ii. 在 Output AMI (输出 AMI) 下提供以下规范。
      A. 输入输出 AMI 的 Name (名称)。在映像管道完成后，这将是创建的 AMI 的名称。
      B. 在 AMI tags (AMI 标签) 下，为映像添加 Key (键) 和可选的 Value (值) 标签。有关标记资源的更多信息，请参阅标记 Amazon EC2 资源。
   iii. 在 AMI distribution settings (AMI 分发设置) 下，您可以指定要将 AMI 复制到的其他 AWS 区域。您也可以配置出站 AMI 的权限。您可以选择允许所有 AWS 账户或仅特定账户启动创建的 AMI。如果您选择允许所有 AWS 账户启动 AMI，则输出 AMI 是公有的。
      A. 选择要分发 AMI 的 AWS 区域。默认情况下，将包括您的当前区域。
      B. 在 Launch permissions (启动权限) 下，您可以将 AMI 设置为 Private (私有) 或 Public (公有)。默认设置为 Private (私有)。在将启动权限设置为私有时，您可以为特定的 AWS 账户授予权限。如果将它们设置为公有，则所有 AWS 用户都有权访问输出 AMI。
         I. 选择 Public (公有) 或 Private (私有)。
         II. 如果选择 Private (私有)，请输入您要授予启动权限的账户的账号，然后选择 Add (添加)。

3. 在 Review and create (检查和创建) 页面上，您可以在创建映像管道之前检查所有设置。检查 Recipe details (配方详细信息)、Pipeline configuration details (管道配置详细信息) 和 Additional settings (其他设置)。如果要进行更改，请选择 Edit (编辑) 以返回到要更改或更新的规范设置。在这些设置反映了所需的配置时，选择 Create Pipeline (创建管道)。
4. 如果映像管道创建失败，您将会收到一条消息，其中包含返回的错误。纠正这些错误，然后再次尝试创建管道。
5. 在成功创建映像管道后，您将转到 Image pipelines (映像管道) 页面。从该页面中，您可以管理、删除、禁用和运行映像管道以及查看有关映像管道的详细信息。

使用 EC2 映像生成器 控制台管理和运行镜像

本节包含帮助您使用 镜像生成器 控制台管理和运行 EC2 映像生成器 镜像的信息。

本节内容

编辑配置详细信息和其他设置

在创建镜像管道后，您可以编辑其配置详细信息和其他设置。要编辑配置详细信息和其他设置，请使用以下步骤。

1. 要编辑配置详细信息（包括描述、生成计划或基础设施详细信息），请导航到 Image pipelines (镜像管道) 页面，然后选中要编辑的管道名称旁边的复选框。接下来，选择 View details (查看详细信息) 按钮。
2. 在 Image pipeline details (镜像管道详细信息) 页面上，查看镜像管道的 Summary (摘要) 详细信息以及 Output image (输出镜像)、Configuration (配置) 和 Image recipe (镜像配方) 选项卡。
3. 在 Configuration (配置) 选项卡中，选择 Configuration details (配置详细信息) 旁边的 Edit (编辑)。
4. 在 Edit configuration details (编辑配置详细信息) 页面上，可以编辑您的配置。然后，选择 Save changes (保存更改)。
5. 要编辑其他设置（包括关联的许可证、AMI 分发设置、镜像导出设置和 SNS 通知设置），请导航到 Image pipelines (镜像管道) 页面，然后选中要编辑的管道名称旁边的复选框。选择 View details (查看详细信息)。

   Note

   不会在您账户中必须启用的 AWS 区域之间复制 License Manager 设置，例如，在 `ap-east-1` (HKG) 和 `me-south-1` (BAH) 区域之间。
6. 在 Image pipeline detail (镜像管道详细信息) 页面上的 Configuration (配置) 选项卡中，选择 Additional settings (其他设置) 旁边的 Edit (编辑)。
7. 在 Edit additional settings (编辑其他设置) 页面上，可以编辑您的配置。然后，选择 Save changes (保存更改)。

在创建镜像配方后，无法修改或替换其组件。如果要更新镜像配方中的组件，请创建新的配方或配方版本。

删除管道

要删除镜像管道，请使用以下步骤。

1. 导航到 Image pipelines (镜像管道) 页面，然后选中要删除的管道名称旁边的复选框。

2. 从 Actions (操作) 下拉列表中，选择 Delete (删除)。
3. 将提示您在文本框中输入 Delete，然后选择 Delete (删除) 以确认删除管道。

在删除管道时，对于与该管道关联的资源，将取消它们的关联并删除配置。

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 镜像管道
2. 基础设施配置/分发配置/镜像配方
3. 组件
4. 镜像

创建新的组件

您可以创建组件以添加到镜像配方中。组件包括生成组件和测试组件。生成组件是用于定义一系列步骤以下载、安装和配置软件包的编排文档。它们还定义了验证和安全强化步骤。测试组件是用于定义要对软件包运行的测试的编排文档。组件是使用 YAML 文档格式定义的。在创建镜像配方后，无法修改或替换其组件。要在创建镜像配方后更新组件，您必须创建新的配方或配方版本。要创建新的组件，请使用以下步骤。

1. 在左侧导航窗格中选择 Components (组件)。然后，选择 Create component (创建组件)。
2. 在 Create component (创建组件) 页面上的 Component details (组件详细信息) 下，输入以下内容：
   - Image Operating system (OS) (镜像操作系统 (OS))。指定与组件兼容的操作系统。
   - Component category (组件类别)。从下拉列表中，选择要创建的生成或测试组件的类型。
   - Component name (组件名称)。输入组件的名称。
   - Component version (组件版本)。输入组件的版本号。
   - Description (描述)。提供可选的描述以帮助您标识组件。
   - Change description (更改描述)。提供可选的描述，以帮助您了解对该组件版本进行的更改。
3. 在 Definition document (定义文档) (这是定义 镜像生成器 对镜像执行的操作的文档) 下，在提供的字段中输入 YAML 格式的文档内容。您可以选择使用 AWS 提供的示例 (在选择 Use example (使用示例) 时自动填充) ，并内联编辑内容。
4. 在输入组件详细信息后，选择 Create component (创建组件)。在创建配方 (p. 13)或创建管道时，将在 Component (组件) 下拉列表中显示该组件。
5. 要删除组件，请从 Components (组件) 页面中选中要删除的组件旁边的复选框。从 Actions (操作) 下拉列表中，选择 Delete component (删除组件)。

您可以选中组件旁边的复选框，然后在 Actions (操作) 下拉列表中选择 Create new version (创建新的版本) 以创建新的组件版本。系统将转到 Create Component (创建组件) 页面，您可以在其中创建新的组件版本。

使用镜像配方

在创建镜像配方后，您可以从 EC2 映像生成器 控制台的 Recipes (配方) 页面中管理该配方。您可以删除镜像配方或创建新的配方版本。

删除镜像配方

要删除镜像配方，请选中镜像配方旁边的复选框，然后在 Actions (操作) 下拉列表中选择 Delete recipe (删除配方)。在删除镜像配方时，对于与该镜像配方关联的镜像和组件，将取消它们的关联并删除配置。将显示一个对话框，提示您输入 delete 以确认删除。

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 镜像管道
2. 基础设施配置/分发配置/镜像配方
3. 组件
4. 镜像

创建新的镜像配方版本

要创建新的镜像配方版本，请选中镜像配方旁边的复选框，然后在 Actions (操作) 下拉列表中选择 Create new version (创建新的版本)。系统将转到 Create Recipe (创建配方) 页面，您可以在其中创建新的镜像配方版本。有关创建镜像配方的说明，请参阅使用 EC2 映像生成器 控制台生成操作系统映像并自动完成部署 (p. 8)中的步骤。

测试

通常，每个测试包含测试脚本、测试二进制文件和测试元数据。测试脚本包含用于启动测试二进制文件的编排命令，可以使用操作系统支持的任何语言编写该测试二进制文件。退出状态代码指示测试结果。测试元数据描述测试及其行为（例如，名称、描述、测试二进制文件路径以及预期的持续时间）。

要使用 EC2 映像生成器 控制台更新镜像配方中的测试，请执行创建新的配方版本 (p. 13)所需步骤，然后更新 Components (组件) 下的 Test Components (测试组件)。

分发

EC2 映像生成器 可以将 AMI 分发到任何 AWS 区域。AMI 将复制到您在用于生成镜像的账户中指定的每个区域。您可以定义 AMI 启动权限，以控制允许哪些 AWS 账户使用创建的 AMI 启动 EC2 实例。例如，您可以将镜像设置为私有、公有或与特定账户共享。如果您将 AMI 分发到其他区域并为其他账户定义启动权限，启动权限将传播到分发了 AMI 的所有区域中的 AMI。

要使用 EC2 映像生成器 控制台更新分发设置，请执行创建新的配方版本 (p. 13)所需步骤，然后在 Configure additional settings (配置其他设置) 页面上的 AMI Distribution settings (AMI 分发设置) 下更新 AWS Regions (AWS 区域) 和/或 Launch permissions (启动权限)。

共享资源

要在账户之间或在 AWS Organizations 中共享组件、镜像配方或镜像，请参阅EC2 映像生成器 中的资源共享 (p. 51)。

合规性

对于 CIS，EC2 映像生成器 使用 Amazon Inspector 自动评估泄露、漏洞以及不符合最佳实践和合规性标准的情况。例如，它会评估不合预期的网络可访问性、未修补的 CVE、公有 Internet 连接和远程根登录支持。Amazon Inspector 是作为测试组件提供的，您可以选择将其添加到镜像配方中。。为了进行强化，EC2 映像生成器 使用 STIG 进行验证。有关通过 镜像生成器 提供的 STIG 组件的完整列表，请参阅 EC2 映像

生成器 STIG 组件 (p. 47)。有关更多信息，请参阅 针对 STIG 合规性的 Amazon EC2 Windows Server AMI。

使用 AWS CLI 设置和管理 EC2 映像生成器 映像管道

您可以使用 AWS CLI 设置、配置和管理映像管道。以下示例 CLI 命令显示了常用的操作和示例文件配置，以帮助您创建和管理映像管道。

> Note
>
> 如果使用 Amazon 提供的资源或共享的资源，但它们不是您的账户拥有的资源，创建具有标签的映像、配方或管道可能会失败。为了避免这种失败，请创建没有标签的资源，然后在创建资源后添加标签。

镜像生成器 支持以下动态标签：

- `- {{imagebuilder:buildDate}}`

  在生成时解析为生成日期/时间。
- `- {{imagebuilder:buildVersion}}`

  解析为生成版本，这是位于 镜像生成器 ARN 末尾的数字。例如，`"arn:aws:imagebuilder:us-west-2:123456789012:component/myexample-component/2019.12.02/1"` 将生成版本显示为 `1`。

本节内容

创建组件文档

设置管道的第一步是，定义一个将执行 AMI 自定义的文档。该文档可以包含生成、验证和测试阶段。有关更多信息，请参阅 文档架构和定义 (p. 34)。

该示例假定我们已将该文档命名为 `component.yaml`。

```
name: 'An_Example_Document'
description: 'This document has a build, validate and test phase'
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download_Scripts
        action: S3Download
        inputs:
          - source: 's3://my-s3-bucket/my-path/my_zip_archive.zip'
            destination: 'c:\mydirectory\my_zip_archive.zip'
      - name: Extract_Tools
        action: ExecutePowerShell
        inputs:
          commands:
            - 'Expand-Archive -LiteralPath
 {{build.Download_Scripts.inputs[0].destination}}'
      - name: 'DisableHibernation'
        action: ExecutePowerShell
        inputs:
          commands:
            - c:\ec2amibuild\scripts\windows\Disable-Hibernation.ps1
  - name: validate
```

```
    steps:
      - name: DiskPercentageFree
        action: ExecutePowerShell
        inputs:
          commands:
            - |
              Function DiskPercentFree {
                [CmdletBinding()]
                Param (
                    [Parameter(Mandatory=$true)]
                    [ValidateNotNullOrEmpty()]
                    [string]$driveLetter,

                    [Parameter(Mandatory=$true)]
                    [ValidateNotNullOrEmpty()]
                    [string]$threshHold
                )
                $disk = Get-PSDrive $driveLetter | Select-Object Used,Free
                $percentage_free = [Math]::round($disk.free/($disk.free+$disk.used) *
100,2)
                if($percentage_free -ge $threshHold) {
                    return 0
                }
                return -1
              }
              DiskPercentFree -driveLetter C -threshHold 10
  - name: test
    steps:
      - name: 'RunTests'
        action: ExecutePowerShell
        inputs:
          commands:
            - c:\ec2amibuild\scripts\windows\TestAMI.ps1
```

将文档上传到 Amazon S3

只有在文档超过 64 KB 时，才需要执行该步骤。在创建 EC2 映像生成器 组件时，可以内联提供较小的文档。超过 64 KB 的文档必须存储在 Amazon S3 中。

```
aws s3 cp component.yaml s3://my-s3-bucket/my-path/component.yaml
```

上传文档引用的所有资源

您必须上传文档引用的所有资源，否则，文档执行将会在运行时失败。

```
aws s3 cp my_zip_archive.zip s3://my-s3-bucket/my-path/my_zip_archive.zip
```

创建组件

接下来，创建一个组件以引用按上述步骤创建的文档。稍后，您将在用于自定义映像的映像配方中引用该组件。

该示例假定我们具有一个名为 `create-component.json` 的文件。

```
{
```

```
    "name": "MyExampleComponent",
    "semanticVersion": "2019.12.02",
    "description": "An example component that builds, validates and tests an image",
    "changeDescription": "Initial version.",
    "platform": "Windows",
    "uri": "s3://my-s3-bucket/my-path/component.yaml",
    "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/60763706-
b131-418b-8f85-3420912f020c",
    "tags": {
        "MyTagKey": "Some Value"
    }
}
```

要创建组件，请使用以下命令。

```
aws imagebuilder create-component --cli-input-json file://create-component.json
```

导入组件

在某些情况下，从预先存在的脚本入手可能更容易一些。对于本文中的情况，您可以执行以下操作。

该示例假定您具有一个名为 import-component.json 的文件（如下所示）。请注意，该文件直接引用名为 AdminConfig.ps1 的 PowerShell 脚本，该脚本已上传到 *my-s3-bucket* 中。目前，组件 format 支持 SHELL。

```
{
    "name": "MyImportedComponent",
    "semanticVersion": "1.0.0",
    "description": "An example of how to import a component",
    "changeDescription": "First commit message.",
    "format": "SHELL",
    "platform": "Windows",
    "type": "BUILD",
    "uri": "s3://my-s3-bucket/AdminConfig.ps1",
    "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/60763706-
b131-418b-8f85-3420912f020c"
}
```

要导入组件，请运行以下命令。

```
aws imagebuilder import-component --cli-input-json file://import-component.json
```

删除组件

以下示例说明了如何指定 ARN 以删除组件生成版本。

```
aws imagebuilder delete-component --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.02/1
```

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 映像管道
2. 基础设施配置/分发配置/映像配方

3. 组件
4. 映像

创建基本映像配方

在创建了组件后，您可以创建映像配方。映像配方定义要作为起点的映像以及用于自定义映像的一组组件。映像配方定义输出映像的内容。以下示例说明了如何使用基本映像配方，这是开始创建的最低配置要求。

该映像配方引用您在前面的步骤中创建的两个组件。您必须将示例中显示的 ARN 替换为您在创建组件时收到的 ARN。对于您的配置，AWS 区域和账户 ID 也是不同的。

Important

组件是按指定顺序安装的。

以下示例引用 Windows Server 2016 英文版完整基本映像。根据您指定的语义版本筛选条件，该示例中的 ARN 引用 SKU 中的最新映像。在该示例中，映像 ARN 为 `arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/2019.x.x`。该 ARN 以 `/2019.x.x` 结尾，这会向 EC2 映像生成器 表明您要使用 2019 年创建的最新 AMI。您可以提供要使用的特定版本，也可以在所有字段中使用通配符。

```
{
    "name": "MyBasicRecipe",
    "description": "This example image recipe creates a Windows 2016 image.",
    "semanticVersion": "2019.12.03",
    "components": [
        {
            "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-component/2019.12.02/1"
        },
        {
            "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-imported-component/1.0.0/1"
        }
    ],
    "parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/2019.x.x"
}
```

假设您在 `create-image-recipe.json` 中存储了映像配方定义，您可以创建映像配方。

```
aws imagebuilder create-image-recipe --cli-input-json file://create-image-recipe.json
```

删除映像配方

以下示例说明了如何指定 ARN 以删除映像配方。

```
aws imagebuilder delete-image-recipe --image-recipe-arn arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2019.12.03
```

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 映像管道
2. 基础设施配置/分发配置/映像配方

3. 组件
4. 映像

创建分发配置

通过使用分发配置，您可以指定输出 AMI 的名称和描述，授权其他 AWS 账户启动 AMI，以及将 AMI 复制到其他 AWS 区域。您还可以将 AMI 导出到 Amazon S3。要公开 AMI，请将启动许可授权账户设置为 `all`。请参阅位于 EC2 ModifyImageAttribute 的公开 AMI 示例。

`create-distribution-configuration.json` 内容如下所示。

```
{
    "name": "MyExampleDistribution",
    "description": "Copies AMI to eu-west-1 and exports to S3",
    "distributions": [
        {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "Name {{imagebuilder:buildDate}}",
                "description": "An example image name with parameter references",
                "amiTags": {
                    "KeyName": "Some Value"
                },
                "launchPermission": {
                    "userIds": [
                        "987654321012"
                    ]
                }
            }
        },
        {
            "region": "eu-west-1",
            "amiDistributionConfiguration": {
                "name": "My {{imagebuilder:buildVersion}} image
 {{imagebuilder:buildDate}}",
                "amiTags": {
                    "KeyName": "Some value"
                },
                "launchPermission": {
                    "userIds": [
                        "100000000001"
                    ]
                }
            }
        }
    ]
}
```

可以使用 JSON 文件创建分发配置。

```
aws imagebuilder create-distribution-configuration --cli-input-json file://create-
distribution-configuration.json
```

更新分发配置

以下示例显示了一个 `update-distribution-configuration.json` 文件以及一个 CLI 命令，该命令用于更新引用该 JSON 文件的分发配置。

示例 update-distribution-configuration.json 内容如下所示。

```
{
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
    "description": "Copies AMI to eu-west-2 and exports to S3",
    "distributions": [
      {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "Name {{imagebuilder:buildDate}}",
                "description": "An example image name with parameter references",
                "launchPermissions": {
                    "userIds": [
                        "987654321012"
                    ]
                }
            }
        },
        {
            "region": "eu-west-2",
            "amiDistributionConfiguration": {
                "name": "My {{imagebuilder:buildVersion}} image
 {{imagebuilder:buildDate}}",
                "tags": {
                    "KeyName": "Some value"
                },
                "launchPermissions": {
                    "userIds": [
                        "100000000001"
                    ]
                }
            }
        }
    ]
}
```

运行以下命令，该命令引用上述 update-distribution-configuration.json 文件。

```
aws imagebuilder update-distribution-configuration --cli-input-json file://update-
distribution-configuration.json
```

删除分发配置

以下示例说明了如何指定 ARN 以删除分发配置。

```
aws imagebuilder delete-distribution-configuration --distribution-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-
distribution-configuration
```

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 映像管道
2. 基础设施配置/分发配置/映像配方
3. 组件

4. 映像

创建基础设施配置

通过使用基础设施配置，您可以指定在其中生成和测试映像的基础设施。在基础设施配置中，您可以指定要与您的实例关联的实例类型、子网和安全组。您也可以将 Amazon EC2 密钥对与用于生成映像的实例相关联。这样，在生成失败并且 `terminateInstanceOnFailure` 设置为 `false` 时，您可以登录到实例以进行故障排除。如果配置日志记录，在基础设施配置中指定的实例配置文件必须具有目标存储桶 (`arn:aws:s3:::BucketName/*`) 的 `s3:PutObject` 权限。

```
{
    "name": "MyExampleInfrastructure",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
        "m5.large", "m5.xlarge"
    ],
    "instanceProfileName": "myIAMInstanceProfileName",
    "securityGroupIds": [
        "sg-12345678"
    ],
    "subnetId": "sub-12345678",
    "logging": {
        "s3Logs": {
            "s3BucketName": "my-logging-bucket",
            "s3KeyPrefix": "my-path"
        }
    },
    "keyPair": "myKeyPairName",
    "terminateInstanceOnFailure": false,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:MyTopic"
}
```

示例基础设施配置存储在名为 `create-infrastructure-configuration.json` 的文件中。

示例配置指定两种实例类型（`m5.large` 和 `m5.xlarge`）。我们建议指定多种实例类型，因为这允许 EC2 映像生成器 从具有足够容量的池中启动实例。这可以减少临时的生成失败次数。

实例配置文件名称用于为实例提供执行自定义活动所需的权限。例如，如果您具有一个从 Amazon S3 中检索资源的组件，则实例配置文件需要具有访问这些文件的权限。该实例配置文件还需要具备能使 EC2 映像生成器 成功与实例进行通信的最第权限。有关更多信息，请参阅先决条件 (p. 7)。

可以使用 JSON 文件创建基础设施配置。

```
aws imagebuilder create-infrastructure-configuration --cli-input-json file://create-
infrastructure-configuration.json
```

更新基础设施配置

以下示例显示了一个 `update-infrastructure-configuration.json` 文件以及一个 CLI 命令，该命令用于更新引用该 JSON 文件的基础设施配置。

示例 `update-infrastructure-configuration.json` 内容如下所示。

```
{
```

```
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
    "description": "An example that will terminate instances of failed builds",
    "instanceTypes": [
        "m5.large", "m5.2xlarge"
    ],
    "instanceProfileName": "myIAMInstanceProfileName",
    "securityGroupIds": [
        "sg-12345678"
    ],
    "subnetId": "sub-12345678",
    "logging": {
        "s3Logs": {
            "s3BucketName": "my-logging-bucket",
            "s3KeyPrefix": "my-path"
        }
    },
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:MyTopic"
}
```

运行以下命令，该命令引用上述 update-infrastructure-configuration.json 文件。

```
aws imagebuilder update-infrastructure-configuration --cli-input-json file://update-
infrastructure-configuration.json
```

删除基础设施配置

以下示例说明了如何指定 ARN 以删除分发配置。

```
aws imagebuilder delete-infrastructure-configuration --infrastructure-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-
infrastructure-configuration
```

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 映像管道
2. 基础设施配置/分发配置/映像配方
3. 组件
4. 映像

创建映像

有了基本配方和基础设施配置后，您便可以创建映像。

```
aws imagebuilder create-image --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2019.12.03 --infrastructure-
configuration-arn arn:aws:imagebuilder:us-west-2123456789012:infrastructure-
configuration/myexampleinfrastructure
```

要查看您在创建映像时创建的资源，请参阅创建的资源 (p. 6)。

取消映像创建

如果要取消正在生成的映像，您可以使用 cancel-image-creation API。

```
aws imagebuilder cancel-image-creation --image-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-recipe/2019.12.03/1
```

删除映像

以下示例说明了如何指定 ARN 以删除映像生成版本。

```
aws imagebuilder delete-image --image-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-image/2019.12.02/1
```

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 映像管道
2. 基础设施配置/分发配置/映像配方
3. 组件
4. 映像

创建映像管道

映像管道自动完成创建黄金映像的过程。该命令类似于我们在前面的步骤中执行的 create-image 步骤。不过，在这种情况下，您可以使用管道配置 EC2 映像生成器 以定期为您生成新的映像。

生成频率取决于您在管道中配置的计划。计划具有两个属性：scheduleExpression 和 pipelineExecutionStartCondition。scheduleExpression 确定 EC2 映像生成器 每隔多长时间评估一次 pipelineExecutionStartCondition。如果将 pipelineExecutionStartCondition 设置为 EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE，则只在尚未部署的已知更改时，EC2 映像生成器 才会生成新的映像。如果它设置为 EXPRESSION_MATCH_ONLY，则每当 CRON 表达式与当前时间匹配时，都将生成新的映像。

create-image-pipeline.json 内容如下所示。

```
{
    "name": "MyWindows2016Pipeline",
    "description": "Builds Windows 2016 Images",
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
recipe/2019.12.03",
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
    "imageTestsConfiguration": {
        "imageTestsEnabled": true,
        "timeoutMinutes": 60
    },
    "schedule": {
        "scheduleExpression": "cron(0 0 * * SUN)",
```

```
        "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
    },
    "status": "ENABLED"
}
```

可以使用 JSON 文件创建映像管道。

```
aws imagebuilder create-image-pipeline --cli-input-json file://create-image-pipeline.json
```

更新映像管道

以下示例显示了一个 update-image-pipeline.json 文件以及一个 CLI 命令，该命令用于更新引用该 JSON 文件的映像管道。

示例 update-image-pipeline.json 内容如下所示。

```
    {
    "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-
example-pipeline",
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
recipe/2019.12.08",
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
    "imageTestsConfiguration": {
        "imageTestsEnabled": true,
        "timeoutMinutes": 120
    },
    "schedule": {
        "scheduleExpression": "cron(0 0 * * MON)",
        "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
    },
    "status": "DISABLED"
}
```

运行以下命令，该命令引用上述 update-image-pipeline.json 文件。

```
aws imagebuilder update-image-pipeline --cli-input-json file://update-image-pipeline.json
```

删除映像管道

以下示例说明了如何指定 ARN 以删除映像管道。

```
aws imagebuilder delete-image-pipeline --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

Note

要在 镜像生成器 中成功删除资源，您必须按以下顺序删除它们。

1. 映像管道

2. 基础设施配置/分发配置/映像配方
3. 组件
4. 映像

将资源策略应用于组件

您可以将资源策略应用于生成组件，以便为生成组件启用跨账户共享。以下命令为其他账户授予权限，以便在其映像配方中使用您的生成组件。要成功执行该命令，您必须确保与您共享资源的账户有权访问共享的生成组件引用的所有资源，例如，在私有存储库上托管的文件。我们建议您使用 RAM CLI 命令 create-resource-share 来共享资源。如果使用 EC2 映像生成器 CLI 命令 put-component-policy，您还必须使用 RAM CLI 命令 promote-resource-share-created-from-policy，以使资源对与您共享资源的所有委托人可见。有关更多信息，请参阅EC2 映像生成器 中的资源共享 (p. 51)。

```
aws imagebuilder put-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.03/1 --policy
 '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal":
 { "AWS": [ "arn:aws:iam::account-id:user/Alice", "account-id-2" ] }, "Action":
 [ "imagebuilder:GetComponent", "imagebuilder:ListComponents" ], "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-
component/2019.12.03/1" ] } ] }'
```

将资源策略应用于映像配方

您可以将资源策略应用于映像配方，以便为映像配方启用跨账户共享。以下命令为其他账户授予权限，以使用您的映像配方在其账户中创建映像。要成功执行该命令，您必须确保与您共享资源的账户有权访问映像配方引用的所有映像或组件。我们建议您使用 RAM CLI 命令 create-resource-share 来共享资源。如果使用 EC2 映像生成器 CLI 命令 put-image-recipe-policy，您还必须使用 RAM CLI 命令 promote-resource-share-created-from-policy，以使资源对与您共享资源的所有委托人可见。有关更多信息，请参阅EC2 映像生成器 中的资源共享 (p. 51)。

```
aws imagebuilder put-image-recipe-policy --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-image-recipe/2019.12.03 --policy
 '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal":
 { "AWS": [ "arn:aws:iam::account-id:user/Alice", "account-id-2" ] }, "Action":
 [ "imagebuilder:GetImageRecipe", "imagebuilder:ListImageRecipes" ], "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-image-
recipe/2019.12.03" ] } ] }'
```

将资源策略应用于映像

您可以将资源策略应用于映像，以允许其他用户在其映像配方中使用该映像。要成功执行以下命令，您必须确保与您共享资源的账户有权访问基础资源（例如 Amazon EC2 AMI）。我们建议您使用 RAM CLI 命令 create-resource-share 来共享资源。如果使用 EC2 映像生成器 CLI 命令 put-image-policy，您还必须使用 RAM CLI 命令 promote-resource-share-created-from-policy，以使资源对与您共享资源的所有委托人可见。有关更多信息，请参阅EC2 映像生成器 中的资源共享 (p. 51)。

```
aws imagebuilder put-image-policy --image-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-image/2019.12.03/1 --policy '{ "Version":
 "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS":
 [ "arn:aws:iam::account-id:user/Alice", "account-id-2" ] }, "Action":
```

```
 ["imagebuilder:GetImage", "imagebuilder:ListImages"] "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image/2019.12.03/1" ] } ] }
```

手动启动映像管道

以下 CLI 示例命令说明了如何手动启动映像管道。运行该命令将导致管道根据需要创建新的映像。

```
aws imagebuilder start-image-pipeline-execution --image-pipeline-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline
```

要查看在运行生成管道时创建的资源，请参阅创建的资源 (p. 6)。

标记资源

以下示例 CLI 命令说明了如何在 EC2 映像生成器 中添加和标记资源。您必须提供 `resourceArn` 以及要为其应用的标签。

示例 `tag-resource.json` 内容如下所示。

```
{
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-
pipeline",
    "tags": {
        "KeyName": "KeyValue"
    }
}
```

运行以下命令，该命令引用上述 `tag-resource.json` 文件。

```
aws imagebuilder tag-resource --cli-input-json file://tag-resource.json
```

取消标记资源

以下示例 CLI 命令说明了如何从资源中删除标签。您必须提供 `resourceArn` 和键以删除标签。

示例 `untag-resource.json` 内容如下所示。

```
{
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-
pipeline",
    "tagKeys": [
        "KeyName"
    ]
}
```

运行以下命令，该命令引用上述 `untag-resource.json` 文件。

```
aws imagebuilder untag-resource --cli-input-json file://untag-resource.json
```

获取组件详细信息

以下示例说明了如何指定 ARN 以获取组件详细信息。

```
aws imagebuilder get-component --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.02/1
```

获取组件策略详细信息

以下示例说明了如何指定 ARN 以获取组件策略详细信息。

```
aws imagebuilder get-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.02
```

获取分发配置详细信息

以下示例说明了如何指定 ARN 以获取分发配置详细信息。

```
aws imagebuilder get-distribution-configuration --distribution-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-
distribution-configuration
```

获取映像

要检查映像进度，请使用 get-image 操作。get-image 返回有关映像、元数据、当前状态和输出资源（如果可用）的详细信息。

```
aws imagebuilder get-image --image-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-recipe/2019.12.03/1
```

获取映像管道详细信息

以下示例说明了如何指定 ARN 以获取映像管道详细信息。

```
aws imagebuilder get-image-pipeline --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

获取映像策略详细信息

以下示例说明了如何指定 ARN 以获取映像策略详细信息。

```
aws imagebuilder get-image-policy --image-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-image/2019.12.02
```

获取映像配方详细信息

以下示例说明了如何指定 ARN 以获取映像配方详细信息。

```
aws imagebuilder get-image-recipe --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2019.12.03
```

获取映像配方策略详细信息

以下示例说明了如何指定 ARN 以获取映像配方策略详细信息。

```
aws imagebuilder get-image-recipe-policy --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2019.12.03
```

获取基础设施配置详细信息

以下示例说明了如何指定 ARN 以获取基础设施配置详细信息。

```
aws imagebuilder get-infrastructure-configuration --infrastructure-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-
infrastructure-configuration
```

列出组件

以下示例说明了如何列出您有权访问的所有组件语义版本。

```
aws imagebuilder list-components
```

您可以选择筛选您拥有的组件、Amazon 提供的组件或其他账户与您共享的组件以查看相应的组件。默认情况下，该请求仅显示您的账户拥有的组件。

```
aws imagebuilder list-components --owner Self
```

```
aws imagebuilder list-components --owner Amazon
```

```
aws imagebuilder list-components --owner Shared
```

列出组件生成版本

以下示例说明了如何列出具有特定语义版本的组件生成版本。

```
aws imagebuilder list-component-build-versions --component-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-component/2019.12.03
```

列出分发

以下示例说明了如何列出所有分发。

```
aws imagebuilder list-distribution-configurations
```

列出映像

以下示例说明了如何列出您有权访问的所有映像语义版本。

```
aws imagebuilder list-images
```

列出映像生成版本

以下示例说明了如何列出具有特定语义版本的映像生成版本。

```
aws imagebuilder list-image-build-versions --image-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-image/2019.12.03
```

列出映像管道映像

以下示例说明了如何列出特定映像管道创建的所有映像。

```
aws imagebuilder list-image-pipeline-images --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

列出映像管道

以下示例说明了如何列出所有映像管道。

```
aws imagebuilder list-image-pipelines
```

列出映像配方

以下示例说明了如何列出所有映像配方。

```
aws imagebuilder list-image-recipes
```

列出基础设施配置

以下示例说明了如何列出所有基础设施配置。

```
aws imagebuilder list-infrastructure-configurations
```

列出特定资源的所有标签

以下示例说明了如何列出特定资源的所有标签。

```
aws imagebuilder list-tags-for-resource --resource-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

EC2 映像生成器 组件管理器

镜像生成器 使用组件管理应用程序以编排复杂的工作流程，修改系统配置以及测试系统，而无需编写代码。该应用程序使用声明性的文档架构。由于它是一个单独的应用程序，因此，它不需要进行额外的服务器设置。它可以在任何云基础设施和本地运行。

EC2 映像生成器 使用该应用程序执行所有实例上的活动，例如生成、验证和测试。您定义一个文档以描述如何生成、验证和测试镜像。EC2 映像生成器 将组件发送到实例，该应用程序执行定义的阶段、步骤和操作以解释组件并将其应用于实例。在完成后，该应用程序向 EC2 映像生成器 发送摘要。如果在管道配置中指定了 S3 存储桶，它还会将详细的执行输出发送到 Amazon S3。然后，EC2 映像生成器 使用 AWS 强化和清理映像的最佳实践。

- 生成阶段。修改镜像。例如，您可以配置镜像以安装应用程序或修改操作系统防火墙设置。在创建镜像之前，将执行验证阶段以作为生成阶段的一部分。
- 测试阶段。在创建新的镜像后，将对该镜像执行测试。

EC2 映像生成器 按如下方式使用组件管理应用程序。

1. 您定义一个 EC2 映像生成器 组件，它是一个描述如何生成、验证和测试镜像的文档。
2. EC2 映像生成器 将文档和该应用程序复制到实例以分派要执行的工作。
3. 该应用程序执行文档中定义的阶段、步骤和操作。

本节内容

在 EC2 映像生成器 中使用文档

要生成组件，您必须提供基于 YAML 的文档，该文档表示用于创建组件的阶段和步骤。

主题

文档部分

文档的部分如下所示。

- 阶段。阶段是步骤的逻辑分组形式。
  - 每个阶段名称在文档中必须是唯一的。

  - 您可以在文档中定义多个阶段。
  - 镜像生成器 在镜像生成管道中执行称为“生成”、“验证”和“测试”的阶段。
- 步骤。步骤是组成每个阶段的工作流程的各个工作单元。
  - 每个步骤必须定义要执行的操作。
  - 对于每个阶段，每个步骤必须具有唯一的名称。
  - 步骤按顺序执行。
  - 可以将步骤的输入和输出作为后续步骤的输入（“链”）。
  - 每个步骤使用一个操作模块，该模块返回一个退出代码。
- 支持的操作。支持的操作是每个步骤必须在文档中包含的操作。每个支持的操作与一个操作模块相关。有关支持的操作模块的完整列表（包括功能详细信息以及输入/输出值和示例），请参阅组件管理器支持的操作模块 (p. 38)。
- 输出文件。配置管理应用程序为每次执行创建以下输出文件：
  - detailedOutput.json：描述有关编排的所有详细信息的文件。包含有关执行的每个阶段、步骤和操作的信息。
  - document.yaml：发送到该应用程序以执行的文件。存储为执行的构件。
  - console.log：包含在执行期间捕获的所有标准输出 (stdout) 和标准错误 (stderr) 信息。
  - application.log：包含调试执行生成的日志。

输入和输出链

配置管理应用程序提供了一种功能，它使用以下格式编写引用以将输入和输出链在一起：

```
{{ phase_name.step_name.inputs/outputs.variable }}
```

或

```
{{ phase_name.step_name.inputs/outputs[index].variable }}
```

通过使用链功能，您可以回收代码并提高文档的可维护性。

链的使用要求如下所示：

- 只能在每个步骤的输入部分中使用链表达式。
- 具有链表达式的语句必须用引号引起来。例如：
  - 无效的表达式：`echo {{ phase.step.inputs.variable }}`
  - 有效的表达式：`"echo {{ phase.step.inputs.variable }}"`
  - 有效的表达式：`'echo {{ phase.step.inputs.variable }}'`
- 链表达式可以引用同一文档中的其他步骤和阶段的变量。
- 链表达式中的索引从 0 开始编制（第一个索引为 0）。

示例

要在以下示例步骤的第二个条目中引用源变量，链模式为 `{{ build.SampleS3Download.inputs[1].source }}`。

```
phases:
  -
    name: 'build'
    steps:
      -
        name: SampleS3Download
```

```
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          -
            source: 's3://sample-bucket/sample1.ps1'
            destination: 'C:\Temp\sample1.ps1'
          -
            source: 's3://sample-bucket/sample2.ps1'
            destination: 'C:\Temp\sample2.ps1'
```

要引用以下示例步骤的输出变量（等于“Hello”），链模式为 {{ build.*SamplePowerShellStep*.outputs.stdout }}。

```
phases:
  -
    name: 'build'
    steps:
      -
        name: SamplePowerShellStep
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          commands:
            - 'echo "Hello"'
```

文档架构和定义

以下是文档的 YAML 架构。

```
name: (optional)
description: (optional)
schemaVersion: "string"

phases:
  - name: "string"
    steps:
        - name: "string"
          action: "string"
          timeoutSeconds: integer
          onFailure: "Abort|Continue"
          maxAttempts: integer
          inputs:
```

文档的架构定义如下所示。

字段	描述	类型	必需
name	文档的名称。	字符串	否
description	文档的描述。	字符串	否
schemaVersion	文档的架构版本，当前为 1.0。	字符串	是
phases	阶段及其步骤的列表。	列表	是

阶段的架构定义如下所示。

字段	描述	类型	必需
name	阶段的名称。	字符串	是
steps	阶段中的步骤列表。	列表	是

步骤的架构定义如下所示。

字段	描述	类型	必需	默认值
name	用户定义的步骤名称。	字符串		
操作	与执行步骤的模块相关的关键字。	字符串		
timeoutSeconds	在失败/重试之前，步骤运行的秒数。 还支持 -1 值，它表示无限超时。不允许使用 0 和其他负值。	整数	是	7,200 秒（120 分钟）
onFailure	失败时的执行决定。该步骤可能会中止或继续执行下一步。	字符串	是	Abort
maxAttempts	在步骤失败之前允许的最大尝试次数。	整数	否	1
inputs	包含操作模块执行步骤所需的参数。	字典	是	

文档示例架构

以下是一个示例文档架构，用于安装所有可用的 Windows 更新，执行配置脚本，在创建 AMI 之前验证更改以及在创建 AMI 后测试更改。

```
name: RunConfig_UpdateWindows
description: 'This document will install all available Windows updates and execute a config
 script.  It will then validate the changes before an AMI is created.  Then after AMI
 creation, it will test all the changes.'
schemaVersion: 1.0
phases:
  -
    name: build
    steps:
      -
        name: DownloadConfigScript
        action: S3Download
        timeoutSeconds: 60
```

```
      onFailure: Abort
      maxAttempts: 3
      inputs:
        -
          source: 's3://customer-bucket/config.ps1'
          destination: 'C:\Temp\config.ps1'
    -
      name: RunConfigScript
      action: ExecutePowerShell
      timeoutSeconds: 120
      onFailure: Abort
      maxAttempts: 3
      inputs:
        commands:
          - '{{build.DownloadConfigScript.inputs[0].destination}}'
    -
      name: Cleanup
      action: ExecutePowerShell
      timeoutSeconds: 120
      onFailure: Abort
      maxAttempts: 3
      inputs:
        commands:
          - 'Remove-Item {{build.DownloadConfigScript.inputs[0].destination}}'
    -
      name: RebootAfterConfigApplied
      action: Reboot
      inputs:
        delaySeconds: 60
    -
      name: InstallWindowsUpdates
      action: UpdateOS
-
  name: validate
  steps:
    -
      name: DownloadTestConfigScript
      action: S3Download
      timeoutSeconds: 60
      onFailure: Abort
      maxAttempts: 3
      inputs:
        -
          source: 's3://customer-bucket/testConfig.ps1'
          destination: 'C:\Temp\testConfig.ps1'
    -
      name: ValidateConfigScript
      action: ExecutePowerShell
      timeoutSeconds: 120
      onFailure: Abort
      maxAttempts: 3
      inputs:
        commands:
          - '{{validate.DownloadTestConfigScript.inputs[0].destination}}'
    -
      name: Cleanup
      action: ExecutePowerShell
      timeoutSeconds: 120
      onFailure: Abort
      maxAttempts: 3
      inputs:
        commands:
          - 'Remove-Item {{validate.DownloadTestConfigScript.inputs[0].destination}}'
-
  name: test
  steps:
```

```
      -
        name: DownloadTestConfigScript
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          -
            source: 's3://customer-bucket/testConfig.ps1'
            destination: 'C:\Temp\testConfig.ps1'
      -
        name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          commands:
            - '{{test.DownloadTestConfigScript.inputs[0].destination}}'
```

以下是下载和执行自定义 Linux 二进制文件的示例文档架构。

```
name: LinuxBin
description: Download and execute a custom Linux binary file.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://mybucket/myapplication
            destination: /tmp/myapplication
      - name: Enable
        action: ExecuteBash
        onFailure: Continue
        inputs:
          commands:
            - 'chmod u+x {{ build.Download.inputs[0].destination }}'
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: '{{ build.Download.inputs[0].destination }}'
          arguments:
            - '--install'
      - name: Delete
        action: ExecuteBash
        inputs:
          commands:
            - 'rm {{ build.Download.inputs[0].destination }}'
```

以下是使用安装文件安装 AWS CLI 的示例文档架构。

```
name: InstallCLISetUp
description: Install AWS CLI using the setup file
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
```

```
        - source: s3://aws-cli/AWSCLISetup.exe
          destination: C:\Windows\temp\AWSCLISetup.exe
    - name: Install
      action: ExecuteBinary
      onFailure: Continue
      inputs:
        path: '{{ build.Download.inputs[0].destination }}'
        arguments:
          - '/install'
          - '/quiet'
          - '/norestart'
    - name: Delete
      action: ExecutePowerShell
      inputs:
        commands:
          - Remove-Item -Path '{{ build.Download.inputs[0].destination }}' -Force
```

以下是使用 MSI 安装程序安装 AWS CLI 的示例文档架构。

```
name: InstallCLIMSI
description: Install AWS CLI using the MSI installer
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://aws-cli/AWSCLI64PY3.msi
            destination: C:\Windows\temp\AWSCLI64PY3.msi
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: 'C:\Windows\System32\msiexec.exe'
          arguments:
            - '/i'
            - '{{ build.Download.inputs[0].destination }}'
            - '/quiet'
            - '/norestart'
      - name: Delete
        action: ExecutePowerShell
        inputs:
          commands:
            - Remove-Item -Path '{{ build.Download.inputs[0].destination }}' -Force
```

组件管理器支持的操作模块

本节包含在 EC2 映像生成器 配置生成映像的实例时使用的组件管理应用程序支持的操作模块列表。还包括操作模块的相应功能详细信息和输入/输出值。

> **Note**
>
> 所有操作模块都使用与 SSM 代理相同的账户执行，该账户在 Linux 上为 `root`，在 Windows 上为 `NT Authority\SYSTEM`。

操作模块

ExecuteBinary

ExecuteBinary 模块用于执行包含命令行参数列表的二进制文件。

如果执行退出并显示退出代码 `194` (Linux) 或 `3010` (Windows)，则 ExecuteBinary 模块处理系统重新启动。在触发后，该应用程序执行以下操作之一：

- 如果是由 SSM 代理执行的，该应用程序将向调用方返回退出代码。SSM 代理处理系统重新引导并重新调用执行，如从脚本中重新引导托管实例中所述。
- 该应用程序保存当前 `executionstate`，配置重新启动触发器以重新执行该应用程序，然后重新引导系统。

在系统重新启动后，该应用程序执行触发重新启动的相同步骤。如果需要使用该功能，您必须编写幂等脚本以处理多次调用同一 Shell 命令的操作。

输入

原语	描述	类型	必需
`path`	要执行的二进制文件的路径。	字符串	是
`arguments`	包含在执行二进制文件时使用的命令行参数列表。	字符串列表	否

输入示例

```
name: "InstallDotnet"
action: ExecuteBinary
inputs:
  path: C:\PathTo\dotnet_installer.exe
  arguments:
    - /qb
    - /norestart
```

输出

字段	描述	类型
`stdout`	命令执行的标准输出。	字符串

输出示例

```
{
```

```
      "stdout": "success"
}
```

ExecuteBash

ExecuteBash 模块用于通过内联 Shell 代码/命令运行 bash 脚本。该模块支持 Linux。

您在命令块中指定的所有命令和指令将转换为文件（例如 `input.sh`）并使用 bash Shell 执行。Shell 文件的执行结果是该步骤的退出代码。

如果执行退出并且退出代码为 `194`，则 ExecuteBash 模块处理系统重新启动。在触发后，该应用程序执行以下操作之一：

- 如果是由 SSM 代理执行的，该应用程序将向调用方返回退出代码。SSM 代理处理系统重新引导并重新调用执行，如从脚本中重新引导托管实例中所述。
- 该应用程序保存当前 `executionstate`，配置重新启动触发器以重新执行该应用程序，然后重新引导系统。

在系统重新启动后，该应用程序执行触发重新启动的相同步骤。如果需要使用该功能，您必须编写幂等脚本以处理多次调用同一 Shell 命令的操作。

输入

原语	描述	类型	必需
`commands`	包含根据 bash 语法执行的指令或命令列表。允许使用多行 YAML。	列表	是

输入示例

```
name: InstallAndValidateCorretto
action: ExecuteBash
inputs:
  commands:
    - sudo yum install java-11-amazon-corretto-headless -y
    - |
      function fail_with_message() {
          1>&2 echo $1
          exit 1
      }

      ARCH=`/usr/bin/arch`

      JAVA_PATH=/usr/lib/jvm/java-11-amazon-corretto.$ARCH/bin/java
      if [ -x $JAVA_PATH ]; then
          echo "Amazon Corretto 11 JRE is installed."
      else
          fail_with_message "Amazon Corretto 11 JRE is not installed. Failing."
      fi

      JAVAC_PATH=/usr/lib/jvm/java-11-amazon-corretto.$ARCH/bin/javac
      if [ -x $JAVAC_PATH ]; then
          echo "Amazon Corretto 11 JDK is installed."
      else
          fail_with_message "Amazon Corretto 11 JDK is not installed. Failing."
      fi
```

输出

字段	描述	类型
stdout	命令执行的标准输出。	字符串

输出示例

```
{
    “stdout”: “This is the standard output from the shell execution\n"
}
```

如果您执行重新引导，并返回退出代码 194 以作为操作模块的一部分，生成将在启动重新引导的同一操作模块步骤处继续执行。如果执行重新引导而没有退出代码，生成过程可能会失败。

ExecutePowerShell

ExecutePowerShell 模块用于通过内联 Shell 代码/命令运行 PowerShell 脚本。该模块支持 Windows 平台和 Windows PowerShell。

在命令块中指定的所有命令/指令将转换为脚本文件（例如 `input.ps1`）并使用 Windows PowerShell 执行。Shell 文件执行结果是退出代码。

如果 Shell 命令退出并且退出代码为 `3010`，则 ExecutePowerShell 模块处理系统重新启动。在触发后，该应用程序执行以下操作之一：

- 如果是由 SSM 代理执行的，则向调用方返回退出代码。SSM 代理处理系统重新引导并重新调用执行，如从脚本中重新引导托管实例中所述。
- 保存当前 `executionstate`，配置重新启动触发器以重新执行该应用程序，然后重新引导系统。

在系统重新启动后，该应用程序执行触发重新启动的相同步骤。如果需要使用该功能，您必须编写幂等脚本以处理多次调用同一 Shell 命令的操作。

输入

原语	描述	类型	必需
commands	包含根据 PowerShell 语法执行的指令或命令列表。允许使用多行 YAML。	字符串列表	是 必须指定 `commands` 或 `file`，但不能同时指定。
file	包含 PowerShell 脚本文件的路径。PowerShell 将使用 `-file` 命令行参数执行该文件。路径必须指向 `.ps1` 文件。	字符串	是 必须指定 `commands` 或 `file`，但不能同时指定。

输入示例

```
name: InstallMySoftware
action: ExecutePowerShell
inputs:
```

```
  commands:
      - Set-SomeConfiguration -Value 10
      - Write-Host 'Successfully set the configuration.'

name: ExecuteMyScript
action: ExecutePowerShell
inputs:
  file: 'C:\PathTo\MyScript.ps1'
```

输出

字段	描述	类型
stdout	命令执行的标准输出。	字符串

输出示例

```
{
    “stdout”: “This is the standard output from the shell execution\n"
}
```

如果您执行重新引导，并返回退出代码 3010 以作为操作模块的一部分，生成将在启动重新引导的同一操作模块步骤处继续执行。如果执行重新引导而没有退出代码，生成过程可能会失败。

Reboot

Reboot 操作模块重新引导实例。它具有一个可配置的选项以延迟启动重新引导。由于重新引导实例，它不支持步骤超时值。默认行为是 `delaySeconds` 为 `60`，这意味着没有延迟。

如果该应用程序是由 SSM 代理调用的，则向 SSM 代理返回退出代码（Windows 为 `3010`，Linux 为 `194`）。SSM 代理处理系统重新引导，如从脚本中重新引导托管实例中所述。

如果该应用程序是在主机上作为单独进程调用的，它将保存当前执行状态，配置重新引导后自动运行触发器以重新执行该应用程序，然后重新引导系统。

重新引导后自动运行触发器：

- Windows。创建一个包含 At `SystemStartup` 触发器的任务计划程序条目。
- Linux。在 crontab 中添加一个作业。

```
@reboot /download/path/awstoe run --document s3://bucket/key/doc.yaml
```

在该应用程序启动时，将清除该触发器。

要使用 Reboot 操作模块，对于包含重新引导 `exitcode` 的步骤（例如，`3010`），您必须以 `sudo user` 形式运行应用程序二进制文件。

输入

原语	描述	类型	必需	默认值
delaySeconds	在启动重新引导之前延迟特定的时间。	整数	否	0

输入示例

```
name: RebootStep
action: Reboot
onFailure: Abort
maxAttempts: 2
inputs:
    delaySeconds: 60
```

输出

无。

在 Reboot 模块完成后，镜像生成器 继续执行生成中的下一步。

UpdateOS

UpdateOS 操作模块增加了对安装 Windows 和 Linux 更新的支持。

默认情况下，UpdateOS 操作模块安装所有可用的更新。您可以提供一个包括列表（要包括在安装中的一个或多个更新）和/或排除列表（要从安装中排除的一个或多个更新）以覆盖该操作。

如果同时提供了“包括”和“排除”列表，则结果更新列表只能包含在“包括”列表中列出并且在“排除”列表中未列出的那些更新。

- Windows。更新是从目标计算机上配置的更新源中安装的。
- Linux。该应用程序检查 Linux 平台中支持的软件包管理器，并使用 `yum` 或 `apt-get` 软件包管理器。如果不支持这两个软件包管理器，则会返回错误。您应该具有运行 UpdateOS 操作模块的 `sudo` 权限。如果您没有 `sudo` 权限，则会返回 `error.Input`。

输入

原语	描述	类型	必需
`include`	对于 Windows，您可以指定一个或多个 Microsoft 知识库 (KB) 文章 ID 以包括在可以安装的更新列表中。有效的格式为：`KB1234567` 或 `1234567`。 对于 Linux，您可以指定一个或多个要包括在安装列表中的软件包。	字符串列表	否
`exclude`	对于 Windows，您可以指定一个或多个 Microsoft 知识库 (KB) 文章 ID 以包括在要从安装中排除的更新列表中。 对于 Linux，您可以指定一个或多个要从安装中排除的软件包。	字符串列表	否

输入示例

```
name: UpdateMyLinux
action: UpdateOS
onFailure: Abort
maxAttempts: 3
inputs:
    exclude:
        - ec2-hibinit-agent
```

输出

无。

S3Upload

S3Upload 操作模块用于将文件从源文件/文件夹上传到某个 Amazon S3 位置。允许将通配符与源一起使用，并以 * 字符表示通配符。

如果递归 S3Upload 操作失败，将保留已上传的 Amazon S3 文件。

支持的使用案例：

- 将本地文件上传到 S3 对象。
- 将文件夹中的本地文件（使用通配符）上传到 S3 `KeyPrefix`。
- 将本地文件夹（必须将 `recurse` 设置为 `true`）复制到 S3 `KeyPrefix`。

IAM 要求

与实例配置文件关联的 IAM 角色必须具有运行 S3Upload 操作模块的权限。必须将以下 IAM 角色策略附加到与实例配置文件关联的 IAM 角色：针对存储桶/对象（例如 `arn:aws:s3:::BucketName/*`）的 `s3:PutObject`。

输入

原语	描述	类型	必需	默认值
`source`	本地路径。源支持 * 表示的通配符。	字符串	是	
`destination`	远程路径。	字符串	是	
`recurse`	如果设置为 `true`，则递归执行 S3Upload。	字符串	否	`false`

输入示例：将本地文件复制到 S3 对象

以下示例说明了如何将本地文件复制到 Amazon S3 对象。

```
name: MyS3UploadFile
action: S3Upload
onFailure: Abort
maxAttempts: 3
inputs:
    - source: C:\myfolder\package.zip
      destination: s3://mybucket/path/to/package.zip
```

输入示例：将本地文件夹中的所有文件复制到具有 KeyPrefix 的 S3 存储桶

以下示例说明了如何将本地文件夹中的所有文件复制到具有 KeyPrefix 的 Amazon S3 存储桶。该示例不会复制子文件夹或其内容，因为未指定 `recurse`，并且它默认为 `false`。

```
name: MyS3UploadMultipleFiles
action: S3Upload
onFailure: Abort
maxAttempts: 3
inputs:
    - source: C:\myfolder\*
      destination: s3://mybucket/path/to/
```

输入示例：将所有文件和文件夹从本地文件夹递归复制到 S3 存储桶

以下示例说明了如何将所有文件和文件夹从本地文件夹递归复制到具有 KeyPrefix 的 Amazon S3 存储桶。

```
name: MyS3UploadFolder
action: S3Upload
onFailure: Abort
maxAttempts: 3
inputs:
    - source: C:\myfolder\*
      destination: s3://mybucket/path/to/
      recurse: true
```

输出

无。

S3Download

S3Download 操作模块用于将 Amazon S3 对象或 KeyPrefix 下载到本地目标路径。目标路径可能是文件或文件夹。如果目标路径已存在，S3Download 将失败，除非 `override` 设置为 `true`。

如果 S3KeyPrefix 的 S3Download 操作失败，目标文件夹将保留失败时的状态。文件夹内容不会回滚到失败之前的内容。

支持的使用案例：

- 将 S3 对象下载到本地文件。
- 将 S3 对象（在 S3 文件路径中具有 KeyPrefix）下载到本地文件夹，这会将 KeyPrefix 中的所有 S3 文件递归复制到本地文件夹。

IAM 要求

与实例配置文件关联的 IAM 角色必须具有运行 S3Download 操作模块的权限。必须将以下 IAM 角色策略附加到与实例配置文件关联的 IAM 角色：

- 单个文件：针对存储桶/对象（例如 `arn:aws:s3:::BucketName/*`）的 `s3:GetObject`。
- 多个文件：针对存储桶/对象（例如 `arn:aws:s3:::BucketName`）的 `s3:ListBucket` 以及针对存储桶/对象（例如 `arn:aws:s3:::BucketName/*`）的 `s3:GetObject`。

输入

原语	描述	类型	必需
source	远程路径。源支持 * 表示的通配符。	字符串	是

原语	描述	类型	必需
destination	本地路径。	字符串	是

Note

对于以下示例，可以将 Windows 文件夹路径替换为 Linux 路径。例如，可以将 C:*myfolder*\package.zip 替换为 /*myfolder*/package.zip。

输入示例：将 S3 对象复制到本地文件

以下示例说明了如何将 S3 对象复制到本地文件。

```
name: DownloadMyFile
action: S3Download
inputs:
    - source: s3://mybucket/path/to/package.zip
      destination: C:\myfolder\package.zip
```

输入示例：将具有 KeyPrefix 的 S3 存储桶中的所有 S3 对象复制到本地文件夹

以下示例说明了如何将具有 KeyPrefix 的 Amazon S3 存储桶中的所有 S3 对象复制到本地文件夹。S3 没有文件夹概念，因此，将复制与 KeyPrefix 匹配的所有对象。最大对象数限制为 1000 个。

```
name: MyS3DownloadKeyprefix
action: S3Download
maxAttempts: 3
inputs:
    - source: s3://mybucket/path/to/*
      destination: C:\myfolder\
```

输出

无。

SetRegistry

SetRegistry 操作模块接受输入列表，用于设置指定注册表项的值。如果注册表项不存在，则会在定义的路径中创建该注册表项。该功能仅适用于 Windows。

输入

原语	描述	类型	必需
path	注册表项的路径。	字符串	是
name	注册表项的名称。	字符串	是
value	注册表项的值。	字符串/数字/数组	是
type	注册表项的值类型。	字符串	是

支持的路径前缀

- HKEY_CLASSES_ROOT / HKCR:
- HKEY_USERS / HKU:
- HKEY_LOCAL_MACHINE / HKLM:

- HKEY_CURRENT_CONFIG / HKCC:
- HKEY_CURRENT_USER / HKCU:

支持的类型

- BINARY
- DWORD
- QWORD
- SZ
- EXPAND_SZ
- MULTI_SZ

输入示例

```
name: SetRegistryKeyValues
action: SetRegistry
maxAttempts: 3
inputs:
    - path: HKLM:\SOFTWARE\MySoftWare
      name: MyName
      value: FirstVersionSoftware
      type: SZ
    - path: HKEY_CURRENT_USER\Software\Test
      name: Version
      value: 1.1
      type: DWORD
```

输出

无.

EC2 映像生成器 STIG 组件

安全技术实施指南 (STIG) 是 Defense Information Systems Agency (DISA) 创建的配置标准，用于保护信息系统和软件。为使您的系统符合 STIG 标准，您必须安装、配置和测试多种安全设置。

EC2 映像生成器 提供 STIG 组件以帮助您快速生成符合 STIG 标准的映像。这些 STIG 组件扫描错误的配置并运行修复脚本。符合 STIG 要求的组件在来自国防部 (DoD) 的 Windows AMI 上安装 InstallRoot 以安装和更新 DoD 证书，并删除不必要的证书以满足 STIG 合规性要求。使用符合 STIG 要求的组件不会收取额外的费用。

法规遵从性级别

- 高（第一类）

  最严重的风险，包括可能导致机密性、可用性或完整性丢失的任何漏洞。
- 中等（第二类）

  任何可能导致机密性、可用性或完整性丢失的漏洞。可以减轻这些风险。
- 低（第三类）

  任何会降级用于防止机密性、可用性或完整性丢失的措施的漏洞。

主题

Windows STIG 组件

Windows STIG 组件是为单独服务器设计的，并应用本地组策略。

STIG-Build-Windows-Low Version 1.0.1

由于组织特定的策略和/或技术限制，尚未 应用以下 STIG 设置。所有其他适用的 STIG 已应用。有关完整列表，请参阅 STIG 文档库。有关如何查看完整列表的说明，请参阅如何查看 SRG 和 STIG。

- Windows Server 2019 STIG V1 发行版 3：

  V-93149、V-93187、V-93229 和 V-93231
- Windows Server 2016 STIG V1 发行版 11：

  V-73307、V-73649、V-90355、V-90357
- Windows Server 2012R2 STIG V2 发行版 18：

  V-1076、V-1112、V-3472、V-4445、V-26359、V-36678、V-36733、V-40172 和 V-40173
- Microsoft .NET Framework STIG 4.0 V1 发行版 9：

  V-30937 和 V-30972
- Windows 防火墙 STIG V1 发行版 7：

  所有 STIG 设置已应用。
- Internet Explorer 11 STIG V1 发行版 14：

  所有 STIG 设置已应用。

STIG-Build-Windows-Medium Version 1.0.1

由于组织特定的策略和/或技术限制，尚未 应用以下 STIG 设置。所有其他适用的 STIG 已应用。有关完整列表，请参阅 STIG 文档库。有关如何查看完整列表的说明，请参阅如何查看 SRG 和 STIG。

- Windows Server 2019 STIG V1 发行版 3

  V-92975、V-92977、V-93047、V-93049、V-93061、V-93077、V-93147、V-93149、V-93183、V-93185、V-93187、和 V-93571
- Windows Server 2016 STIG V1 发行版 12

  V-73223、V-73229、V-73231、V-73233、V-73235、V-73245、V-73259、V-73261、V-73263、V-73265、V-73273、
- Windows Server 2012R2 STIG V2 发行版 18

  V-1072、V-1076、V-1089、V-1112、V-1114、V-1115、V-1145、V-2907、V-3289、V-3383、V-3472、V-3487、V-44 和 V-75915
- Microsoft .NET Framework STIG 4.0 V1 发行版 9

  V-7055、V-7061、V-7063、V-7067、V-7069、V-7070、V-18395、V-30926、V-30935、V-30937、V-30968、V-3097 和 V-32025
- Windows 防火墙 STIG V1 发行版 7

  所有 STIG 设置已应用。

- Internet Explorer 11 STIG V1 发行版 14

  所有 STIG 设置已应用。

STIG-Build-Windows-High Version 1.0.1

由于组织特定的策略和/或技术限制，尚未 应用以下 STIG 设置。所有其他适用的 STIG 已应用。有关完整列表，请参阅 STIG 文档库。有关如何查看完整列表的说明，请参阅如何查看 SRG 和 STIG。

- Windows Server 2019 STIG V1 发行版 3

  V-92975、V-92977、V-93047、V-93049、V-93051、V-93057、V-93061、V-93077、V-93147、V-93149、V-93183、和 V-93571
- Windows Server 2016 STIG V1 发行版 12

  V-73217、V-73221、V-73223、V-73225、V-73229、V-73231、V-73233、V-73235、V-73241、V-73245、V-73259、
- Windows Server 2012R2 STIG V2 发行版 18

  V-1072、V-1074、V-1076、V-1089、V-1102、V-1112、V-1114、V-1115、V-1127、V-1145、V-2907、V-3289、V-33和 V-75915
- Microsoft .NET Framework STIG 4.0 V1 发行版 9

  V-7055、V-7061、V-7063、V-7067、V-7069、V-7070、V-18395、V-30926、V-30935、V-30937、V-30968、V-3097和 V-32025
- Windows 防火墙 STIG V1 发行版 7

  所有 STIG 设置已应用。
- Internet Explorer 11 STIG V1 发行版 14

  所有 STIG 设置已应用。

Linux STIG 组件

以下几节包含有关 Linux STIG 组件的信息。

STIG-Build-Linux-Low Version 2.6.0

由于组织特定的策略和/或技术限制，尚未 应用以下 STIG 设置。所有其他适用的 STIG 已应用。有关完整列表，请参阅 STIG 文档库。有关如何查看完整列表的说明，请参阅如何查看 SRG 和 STIG。

RHEL STIG V2 发行版 7

V-72003、V-72059、V-72061、V-72063、V-72069、V-72071、V-72275、V-72281、V-81009、V-81011和 V-81013

STIG-Build-Linux-Medium Version 2.6.0

由于组织特定的策略和/或技术限制，尚未 应用以下 STIG 设置。所有其他适用的 STIG 已应用。有关完整列表，请参阅 STIG 文档库。有关如何查看完整列表的说明，请参阅如何查看 SRG 和 STIG。

RHEL STIG V2 发行版 7

V-71863、V-71897、V-71927、V-71931、V-71933、V-71947、V-71957、V-71965、V-71971、V-71973、V-71975、V-和 V-92255

STIG-Build-Linux-High Version 2.6.0

由于组织特定的策略和/或技术限制，尚未 应用以下 STIG 设置。所有其他适用的 STIG 已应用。有关完整列表，请参阅 STIG 文档库。有关如何查看完整列表的说明，请参阅如何查看 SRG 和 STIG。

RHEL STIG V2 发行版 7

V-71855、V-71863、V-71897、V-71927、V-71931、V-71933、V-71937、V-71939、V-71947、V-71957、V-71965、V-
和 V-94843

EC2 映像生成器 中的资源共享

EC2 映像生成器 与 AWS Resource Access Manager (AWS RAM) 集成在一起，以允许您与任何 AWS 账户之间或通过 AWS Organizations 共享某些资源。可以共享的 EC2 映像生成器 资源如下所示：

- 组件
- 镜像
- 镜像配方

本节提供了一些信息以帮助您共享这些 EC2 映像生成器 资源。

本节内容

在 EC2 映像生成器 中使用共享的组件、镜像和镜像配方

通过使用组件、镜像和镜像配方共享功能，资源所有者可以与其他 AWS 账户之间或在 AWS 组织内部共享软件配置。您可以集中管理资源共享，并定义一组可与您共享配置的账户。

在该模型中，拥有组件、镜像或镜像配方的 AWS 账户（所有者）与其他 AWS 账户（使用者）共享这些资源。使用者可以将共享的组件与其镜像管道相关联，以自动使用共享组件、镜像或镜像配方的更新。

组件、镜像或镜像配方所有者可以与以下实体共享这些资源：

- AWS Organizations 中的所有者组织内部或外部的特定 AWS 账户
- AWS Organizations 中的所有者组织内部的组织单位
- AWS Organizations 中的整个所有者组织

共享组件、镜像和镜像配方的先决条件

要共享组件、镜像或镜像配方，必须满足以下条件：

- 您必须在您的 AWS 账户中拥有该资源。您不能共享已与您共享的资源。
- 不能对该资源进行加密。

- 您必须允许与 AWS Organizations 共享，才能与您的组织或 AWS Organizations 中的组织单位共享这些资源。
- 您有责任确保也与使用者共享这些资源外部的依赖项或在 AWS 外部管理的基础资源。EC2 映像生成器 不管理 EC2 映像生成器 外部的依赖项。

相关服务

AWS Resource Access Manager

组件、镜像和镜像配方共享与 AWS Resource Access Manager (AWS RAM) 集成在一起。AWS RAM 是一项服务，允许您与任何 AWS 账户之间或通过 AWS Organizations 共享您的 AWS 资源。通过使用 AWS RAM，您可以创建资源共享以共享您拥有的资源。资源共享指定要共享的资源以及与您共享这些资源的使用者。使用者可能是单个 AWS 账户、组织单位或 AWS Organizations 中的整个组织。

跨区域共享

共享的组件、镜像和镜像配方只能在指定的 AWS 区域中进行共享。在共享这些资源时，不会在区域之间复制它们。

共享组件、镜像或镜像配方

要共享组件、镜像或镜像配方，您必须将其添加到资源共享中。资源共享是一种 AWS Resource Access Manager 资源，可用于在 AWS 账户之间共享您的资源。资源共享指定要共享的资源以及与您共享这些资源的使用者。要将组件、镜像或镜像配方添加到新的资源共享中，您必须先使用 AWS Resource Access Manager 控制台创建资源共享。

如果您属于 AWS Organizations 中的一个组织，并且在您的组织中启用了共享，将为您的组织中的使用者自动授予共享组件、镜像或镜像配方的访问权限。否则，使用者将会收到加入资源共享的邀请，并在接受邀请后为其授予共享资源的访问权限。

您可以使用 AWS Resource Access Manager 控制台或 AWS CLI 共享您拥有的组件、镜像或镜像配方。

使用 AWS Resource Access Manager 控制台共享您拥有的组件、镜像或镜像配方

请参阅 AWS Resource Access Manager 用户指南中的“创建资源共享”。

使用 AWS CLI 共享您拥有的组件、镜像或镜像配方

使用 `create-resource-share` 命令。

使用基于资源的策略共享组件、映像或映像配方

镜像生成器 服务提供的 `PutImagePolicy`、`PutComponentPolicy` 和 `PutImageRecipePolicy` API 允许您分别为映像、组件和映像配方定义资源策略。AWS RAM 利用这些 API 来定义正确的资源策略，以允许与其共享某个资源的使用者执行涉及此共享资源的 API 调用。为了使 AWS RAM 能够设置正确的策略来共享和取消共享某个资源，资源所有者必须具有 `imagebuilder:put*` 权限。

将共享的组件、镜像或镜像配方取消共享

要取消共享您拥有的共享组件、镜像或镜像配方，您必须将其从资源共享中删除。您可以使用 AWS Resource Access Manager 控制台或 AWS CLI 以执行该操作。

Note

要取消共享组件、镜像或镜像配方，使用者不能具有这些资源的任何依赖项。使用者必须先删除共享资源的任何依赖项，然后所有者才能取消共享这些资源。

使用 AWS Resource Access Manager 控制台取消共享您拥有的共享组件、镜像或镜像配方

请参阅 AWS Resource Access Manager 用户指南中的更新资源共享。

使用 AWS CLI 取消共享您拥有的共享组件、镜像或镜像配方

使用 `disassociate-resource-share` 命令。

查找共享的组件、镜像或镜像配方

所有者和使用者可以使用 AWS CLI 查找共享的组件、镜像和镜像配方。

查找共享的组件

使用 `list-components` 命令。该命令返回您拥有的组件以及与您共享的组件。`get-component` 显示组件所有者的 AWS 账户 ID。

查找共享的镜像

使用 `list-images` 命令。该命令返回您拥有的镜像以及与您共享的镜像。`get-image` 显示镜像所有者的 AWS 账户 ID。

查找共享的镜像配方

使用 `list-image-recipes` 命令。该命令返回您拥有的镜像配方以及与您共享的镜像配方。`get-image-recipe` 显示镜像配方所有者的 AWS 账户 ID。

共享的组件、镜像和镜像配方权限

所有者的权限

所有者不能删除共享的组件、镜像或镜像配方，直到不再共享该资源。所有者不能取消共享这些资源，直到没有使用者依赖它们。

使用者的权限

使用者只能读取组件、镜像或镜像配方。使用者不能以任何方式修改它们；如果这些资源由其他使用者或资源所有者拥有，则使用者不能查看或修改它们。使用者可以在镜像配方中使用共享的组件和镜像以创建黄金镜像。使用者可以使用共享的镜像配方以创建黄金镜像。

计费和计量

使用 EC2 映像生成器 不会产生任何费用。

实例限制

共享的组件、镜像和镜像配方仅计入所有者的相应资源限制。使用者的资源限制不受已与他们共享的资源的影响。

EC2 映像生成器 中的安全性

AWS 的云安全性的优先级最高。作为 AWS 客户，您将从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。

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

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。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 EC2 映像生成器 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。

该文档帮助您了解如何在使用 镜像生成器 时应用责任共担模型。以下主题说明了如何配置 镜像生成器 以实现您的安全性和合规性目标。您还会了解如何使用其他 AWS 服务以帮助您监控和保护 镜像生成器 资源。

主题

EC2 Image Builder 和接口 VPC 终端节点 (AWS PrivateLink)

您可以通过创建接口 VPC 终端节点在 VPC 和 EC2 Image Builder 之间建立私有连接。接口终端节点由 AWS PrivateLink 提供支持，该技术支持您通过私有连接访问 Image Builder API，而无需采用互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。VPC 中的实例即使没有公有 IP 地址也可与 Image Builder API 进行通信。VPC 和 Image Builder 之间的流量不会脱离 Amazon 网络。

每个接口终端节点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

关于 Image Builder VPC 终端节点的注意事项

请务必先查看 Amazon VPC 用户指南中的接口终端节点属性和限制，然后再为 Image Builder 设置接口 VPC 终端节点。

Image Builder 支持从 VPC 调用它的所有 API 操作。

为 Image Builder 创建接口 VPC 终端节点

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface (AWS CLI) 为 Image Builder 服务创建 VPC 终端节点。有关更多信息，请参阅 Amazon VPC 用户指南 中的创建接口终端节点。

使用以下服务名称为 Image Builder 创建 VPC 终端节点：

- com.amazonaws.*region*.imagebuilder

如果为终端节点启用私有 DNS，则可以使用 Image Builder 针对区域的默认 DNS 名称向其发出 API 请求，例如 `imagebuilder.us-east-1.amazonaws.com`。

有关更多信息，请参阅 Amazon VPC 用户指南中的通过接口终端节点访问服务。

为 Image Builder 创建 VPC 终端节点策略

您可以为 VPC 终端节点附加控制对 Image Builder 的访问的终端节点策略。该策略指定以下信息：

- 可执行操作的委托人。
- 可执行的操作。
- 可对其执行操作的资源。

> **Important**
>
> 当将非默认策略应用于 EC2 映像生成器 的接口 VPC 终端节点时，某些失败的 API 请求（例如 `RequestLimitExceeded` 失败请求）可能不会记录到 AWS CloudTrail 或 Amazon CloudWatch。

有关更多信息，请参阅 Amazon VPC 用户指南中的使用 VPC 终端节点控制对服务的访问。

示例：Image Builder 操作的 VPC 终端节点策略

以下是拒绝删除 镜像生成器 映像和组件的权限的 Image Builder 终端节点策略示例。该示例策略还授予执行所有其他 EC2 映像生成器 操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Action": "imagebuilder:*",
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Action": [
            "imagebuilder: DeleteImage"
        ],
        "Effect": "Deny",
        "Resource": "*",
    },
    {
        "Action": [
            "imagebuilder: DeleteComponent"
        ],
        "Effect": "Deny",
        "Resource": "*",
    }]
}
```

EC2 映像生成器 中的数据保护

EC2 映像生成器 符合 AWS 责任共担模型，该模型包含适用于数据保护的法规和准则。AWS 负责保护运行所有 AWS 服务的全球基础设施。AWS 保持对该基础设施上托管的数据的控制，包括用于处理客户内容和个人数据的安全配置控制。作为数据控制者或数据处理者，AWS 客户和 APN 合作伙伴对他们放在 AWS 云中的任何个人数据承担责任。

出于数据保护的目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management (IAM) 设置单个用户账户，以便仅向每个用户提供履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务，例如 Amazon Macie（帮助查找和保护在 Amazon S3 中存储的个人数据）或 Amazon GuardDuty（持续监控恶意活动和未经授权的行为以保护您的 AWS 账户和工作负载）。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如 Name (名称) 字段）。这包括通过控制台、API、AWS CLI 或 AWS 开发工具包使用 镜像生成器 或其他 AWS 服务时。可能会选取您输入到 镜像生成器 或其他服务中的任何数据以包含在诊断日志中。当您向外部服务器提供 URL 时，请勿在 URL 中包含凭证信息来验证您对该服务器的请求。

由 镜像生成器 创建的所有镜像以及用于生成它们的 Amazon EC2 实例都位于您的账户中。因此，在镜像或实例中包含的任何内容不在 镜像生成器 服务范围内。镜像生成器 服务存储您创建的组件定义。您可以上传 镜像生成器 服务中存储的自定义组件定义。自定义组件是使用您的 KMS 密钥或 镜像生成器 拥有的 KMS 密钥加密的。

有关数据保护的更多信息，请参阅 AWS 安全性博客上的 AWS 责任共担模型和 GDPR 博客文章。

EC2 映像生成器 中的加密和密钥管理

默认情况下，镜像生成器 加密传输中的数据和静态数据。可以将该服务中定义的自定义组件添加到您的镜像管道中，并与其他客户账户共享。您无需共享您的组件即可生成镜像。

自定义组件是使用您的 KMS 密钥或 镜像生成器 拥有的 KMS 密钥加密的。镜像生成器 不会在该服务中存储任何日志。所有日志保存在用于生成镜像的 Amazon EC2 实例上或 SSM Automation 日志中。

您可以通过 AWS KMS 管理密钥。您无法管理 镜像生成器 拥有的 镜像生成器 AWS KMS 密钥。

有关使用 AWS Key Management Service 管理 AWS KMS 密钥的更多信息，请参阅 AWS Key Management Service Developer Guide 中的入门。

EC2 映像生成器 中的互联网络流量保密性

可以通过 HTTPS 保护 镜像生成器 和本地位置之间、AWS 区域中的 AZ 之间以及 AWS 区域之间的连接。在账户之间没有直接连接。

适用于 EC2 映像生成器 的 Identity and Access Management

主题

受众

如何使用 AWS Identity and Access Management (IAM) 因您可以在 EC2 Image Builder 中执行的操作而异。

服务用户 – 如果您使用 EC2 Image Builder 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 EC2 Image Builder 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 EC2 Image Builder 中的一项功能，请参阅对 EC2 映像生成器 身份和访问进行故障排除 (p. 66)。

服务管理员 – 如果您在公司负责管理 EC2 Image Builder 资源，则您可能具有 EC2 Image Builder 的完全访问权限。您有责任确定您的员工应访问哪些 EC2 Image Builder 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 EC2 Image Builder 搭配使用的更多信息，请参阅EC2 映像生成器 如何与 IAM 一起使用 (p. 57)。

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 EC2 Image Builder 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 EC2 Image Builder 示例策略，请参阅EC2 映像生成器 基于身份的策略示例 (p. 60)。

使用身份进行身份验证

有关如何为您的 AWS 账户中的人员和流程提供身份验证的详细信息，请参阅 IAM 用户指南 中的身份。

使用策略管理访问

有关如何创建策略并将其附加到 IAM 身份或 AWS 资源以在 AWS 中管理访问的详细信息，请参阅 IAM 用户指南 中的策略和权限。

与实例配置文件关联的 IAM 角色必须有权运行映像中包含的生成和测试组件。必须将以下 IAM 角色策略附加到与实例配置文件关联的 IAM 角色：EC2InstanceProfileForImageBuilder 和 AmazonSSMManagedInstanceCore。

EC2 映像生成器 如何与 IAM 一起使用

在使用 IAM 管理对 镜像生成器 的访问之前，您应了解哪些 IAM 功能可以与 镜像生成器 一起使用。要大致了解 镜像生成器 和其他 AWS 服务如何与 IAM 一起使用，请参阅 IAM 用户指南 中的与 IAM 一起使用的 AWS 服务。

主题

镜像生成器 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及在哪些条件下允许或拒绝操作。镜像生成器 支持特定的操作、资源和条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南 中的 Amazon EC2 Image Builder 的操作、资源和条件键。

操作

镜像生成器 中的策略操作在操作前面使用以下前缀：`imagebuilder:`。策略语句必须包含 `Action` 或 `NotAction` 元素。镜像生成器 定义了一组自己的操作，以描述可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": [
      "imagebuilder:action1",
      "imagebuilder:action2"
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `List` 开头的所有操作，包括以下操作：

```
"Action": "imagebuilder:List*"
```

要查看 镜像生成器 操作列表，请参阅 IAM 用户指南 中的 AWS 服务的操作、资源和条件键。

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。

镜像生成器 实例资源具有以下 ARN。

```
arn:aws:imagebuilder:region:account-id:resource:resource-id
```

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 `i-1234567890abcdef0` 实例，请使用以下 ARN。

```
"Resource": "arn:aws:imagebuilder:us-east-1:123456789012:instance/i-1234567890abcdef0"
```

要指定属于特定账户的所有实例，请使用通配符 (*)。

```
"Resource": "arn:aws:imagebuilder:us-east-1:123456789012:instance/*"
```

无法对特定资源执行某些 镜像生成器 操作，例如，用于创建资源的操作。在这些情况下，您必须使用通配符 (*)。

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

很多 EC2 映像生成器 API 操作涉及多种资源。要在单个语句中指定多个资源，请使用逗号分隔 ARN。

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

条件键

镜像生成器 提供特定于服务的条件键，并支持使用某些全局条件键。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。提供以下特定于服务的条件键。

imagebuilder:CreatedResourceTagKeys

与字符串运算符结合使用。

使用此键可根据请求中是否存在标签键来筛选访问。这允许您通过定义的标签来管理 Image Builder 创建的资源。

可用性 - 此键仅可用于 `CreateInfrastrucutreConfiguration` 和 `UpdateInfrastructureConfiguration` API。

imagebuilder:CreatedResourceTag/<key>

与字符串运算符结合使用。

使用此键可根据附加到 Image Builder 所创建的资源的标签键值对来筛选访问。这允许您通过定义的标签来管理 Image Builder 创建的资源。

可用性 - 此键仅可用于 `CreateInfrastrucutreConfiguration` 和 `UpdateInfrastructureConfiguration` API。

示例

要查看 镜像生成器 基于身份的策略示例，请参阅 EC2 映像生成器 基于身份的策略示例 (p. 60)。

镜像生成器 基于资源的策略

基于资源的策略是 JSON 策略文档，它指定了指定的委托人可以对 镜像生成器 资源执行的操作以及在哪些条件下执行操作。对于组件、映像和映像配方，镜像生成器 支持基于资源的权限策略。基于资源的策略允许您基于资源向其他账户授予使用权限。您也可以使用基于资源的策略，以允许 AWS 服务访问组件、映像和映像配方。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的委托人 。将跨账户委托人添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源位于不同的 AWS 账户中时，还必须授予委托人实体对资源的访问权限。通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的委托人授予访问权限，则不需要额外的基于身份的策略。有关更多信息，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。

基于 镜像生成器 标签的授权

您可以将标签附加到 镜像生成器 资源，或者将请求中的标签传递给 镜像生成器。要基于标签控制访问，您需要使用 `imagebuilder:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。有关标记 镜像生成器 资源的更多信息，请参阅标记资源 (p. 27)。

镜像生成器 IAM 角色

IAM 角色是 AWS 账户中具有特定权限的实体。

将临时凭证用于 镜像生成器

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

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。

镜像生成器 支持服务相关角色。有关创建或管理 镜像生成器 服务相关角色的详细信息，请参阅将服务相关角色用于 EC2 映像生成器 (p. 64)。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在您的 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，这样做可能会中断服务的功能。

EC2 映像生成器 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 镜像生成器 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 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南 中的在 JSON 选项卡上创建策略。

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 EC2 Image Builder 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 EC2 Image Builder，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。

使用 镜像生成器 控制台

要访问 EC2 映像生成器 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 AWS 账户中的 镜像生成器 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（IAM 用户或角色），控制台将无法按预期正常运行。

要确保这些实体仍然可以使用 镜像生成器 控制台，还要将以下 AWS 托管策略之一附加到实体。有关更多信息，请参阅 IAM 用户指南 中的为用户添加权限：

AWSImageBuilderReadOnlyAccess

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "imagebuilder:Get*",
                "imagebuilder:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/imagebuilder.amazonaws.com/
AWSServiceRoleForImageBuilder"
        }
    ]
}
```

AWSImageBuilderFullAccess

```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "imagebuilder:*"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "sns:ListTopics"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "sns:Publish"
                ],
                "Resource": "arn:aws:sns:*:*:*imagebuilder*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "license-manager:ListLicenseConfigurations",
                    "license-manager:ListLicenseSpecificationsForResource"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iam:GetRole"
                ],
                "Resource": "arn:aws:iam::*:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder"
            },
```

```
            {
                "Effect": "Allow",
                "Action": [
                    "iam:GetInstanceProfile"
                ],
                "Resource": "arn:aws:iam::*:instance-profile/*imagebuilder*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iam:ListInstanceProfiles",
                    "iam:ListRoles"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": "iam:PassRole",
                "Resource": [
                    "arn:aws:iam::*:instance-profile/*imagebuilder*",
                    "arn:aws:iam::*:role/*imagebuilder*"
                ],
                "Condition": {
                    "StringEquals": {
                        "iam:PassedToService": "ec2.amazonaws.com"
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:ListAllMyBuckets",
                    "s3:GetBucketLocation"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket"
                ],
                "Resource": "arn:aws:s3::*:*imagebuilder*"
            },
            {
                "Action": "iam:CreateServiceLinkedRole",
                "Effect": "Allow",
                "Resource": "arn:aws:iam::*:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder",
                "Condition": {
                    "StringLike": {
                        "iam:AWSServiceName": "imagebuilder.amazonaws.com"
                    }
                }
            },
            {
                "Effect": "Allow",
                "Action": [
                    "ec2:DescribeImages",
                    "ec2:DescribeVpcs",
                    "ec2:DescribeRegions",
                    "ec2:DescribeVolumes",
                    "ec2:DescribeSubnets",
                    "ec2:DescribeKeyPairs",
                    "ec2:DescribeSecurityGroups"
                ],
                "Resource": "*"
```

```
            }
        ]
    }
```

Important

除了 AWSImageBuilderFullAccess 策略以外，您还必须附加以下自定义策略，并包括希望使用并且在资源名称中不包含“imagebuilder”的资源：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "sns topic arn"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile"
            ],
            "Resource": "instance profile role arn"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "instance profile role arn"
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "bucket arn"
        }
    ]
}
```

对于仅调用 AWS CLI 或 AWS API 的用户，您不需要为其提供最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

EC2 映像生成器 基于资源的策略示例

要了解如何创建组件，请参阅创建新的组件 (p. 12)。

限制 镜像生成器 组件访问特定的 IP 地址

以下示例为任何用户授予权限以对组件执行任何 镜像生成器 操作。但是，请求必须来自条件中指定的 IP 地址范围。

此语句中的条件确定允许的 Internet 协议版本 4 (IPv4) IP 地址范围为 54.240.143.*，只有一个例外：54.240.143.188。

Condition 块使用 IpAddress 和 NotIpAddress 条件以及 aws:SourceIp 条件键（这是 AWS 范围的条件键）。有关这些条件键的更多信息，请参阅在策略中指定条件。aws:sourceIp IPv4 值使用标准 CIDR 表示法。有关更多信息，请参阅 IAM 用户指南 中的 IP 地址条件运算符。

```
{
  "Version": "2012-10-17",
  "Id": "IBPolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "imagebuilder.GetComponent:*",
      "Resource": "arn:aws-cn:imagebuilder:::examplecomponent/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
         "NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}
      }
    }
  ]
}
```

将服务相关角色用于 EC2 映像生成器

EC2 映像生成器 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 镜像生成器 直接相关。服务相关角色是由 镜像生成器 预定义的，并包含该服务代表您调用其他 AWS 服务所需的所有权限。

可以通过服务相关角色轻松设置 镜像生成器，因为您不必手动添加所需的权限。镜像生成器 定义其服务相关角色的权限，除非另行定义，否则，仅 镜像生成器 可以担任其角色。定义的权限包括信任策略和权限策略。不能将该权限策略附加到任何其他 IAM 实体。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的 AWS 服务并查找服务相关角色 列为是 的服务。选择 Yes 与查看该服务的服务相关角色文档的链接。

镜像生成器 的服务相关角色权限

镜像生成器 使用名为 AWSServiceRoleForImageBuilder 的服务相关角色以允许 EC2 映像生成器 代表您访问 AWS 资源。

AWSServiceRoleForImageBuilder 服务相关角色信任以下服务以担任该角色：

- imagebuilder.amazonaws.com
- ssm.amazonaws.com

角色权限策略允许 镜像生成器 服务对指定的资源完成以下操作：

EC2

- ec2:CancelExportTask
- ec2:CopyImage
- ec2:CreateImage
- ec2:CreateLaunchTemplate
- ec2:CreateTags
- ec2:DeleteLaunchTemplate
- ec2:DeregisterImage
- ec2:DescribeImages

- ec2:DescribeInstanceStatus
- ec2:DescribeSubnets
- ec2:DescribeTags
- ec2:ModifyImageAttribute
- ec2:RunInstances
- ec2:StopInstances
- ec2:TerminateInstances

IAM

- iam:CreateServiceLinkedRole

License Manager

- license-manager:UpdateLicenseSpecificationsForResource

SSM

- ssm:AddTagsToResource
- ssm:DescribeInstanceInformation
- ssm:GetAutomationExecution
- ssm:SendCommand
- ssm:StartAutomationExecution
- ssm:StopAutomationExecution

SNS

- sns:Publish

您必须配置权限以允许 IAM 实体 (例如，用户、组或角色) 创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的“服务相关角色权限”。

为 EC2 映像生成器 创建服务相关角色

您无需手动创建服务相关角色。在 AWS 管理控制台、AWS CLI 或 AWS API 中创建第一个 镜像生成器 资源时，镜像生成器 将为您创建服务相关角色。

> **Important**
>
> 如果您删除了此服务相关角色然后需要再次创建它，则可以使用相同的流程在您的账户中重新创建此角色。在创建第一个 EC2 映像生成器 资源时，镜像生成器 将再次为您创建服务相关角色。

您也可以使用 IAM 控制台为 EC2 映像生成器 使用案例创建服务相关角色。在 AWS CLI 或 AWS API 中，请使用 `imagebuilder.amazonaws.com` 服务名称创建一个服务相关角色。有关更多信息，请参阅 IAM 用户指南 中的创建服务相关角色。如果您删除了此服务相关角色，则可以使用此相同过程再次创建角色。

编辑 镜像生成器 的服务相关角色

可以使用 镜像生成器 控制台、AWS CLI 或 AWS API 更改 AWSServiceRoleForImageBuilder 服务相关角色的描述、信任策略或权限策略，包括添加其他策略。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您只能使用 IAM、AWS CLI 或 API 编辑角色描述。有关更多信息，请参阅 IAM 用户指南 中的编辑服务相关角色。

删除 镜像生成器 的服务相关角色

您可以使用 IAM 控制台、AWS CLI 或 AWS API 手动删除服务相关角色。为此，您必须先手动清除服务相关角色的资源，然后才能手动删除它。

> Note
>
> 在尝试删除资源时，如果 镜像生成器 服务正在使用该角色，删除可能会失败。如果发生这种情况，则请等待几分钟后重试。

删除 AWSServiceRoleForImageBuilder 使用的 镜像生成器 资源

1. 等待当前镜像生成过程完成，或者使用 cancel-image-creation API 明确取消这些生成。要在 镜像生成器 控制台上取消镜像生成，请为每个管道选择 Stop Pipeline (停止管道) 操作按钮。
2. 使用 镜像生成器 控制台或 CLI 删除所有管道，或将所有镜像管道的生成计划更改为手动。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、AWS CLI 或 AWS API 删除 AWSServiceRoleForImageBuilder 服务相关角色。有关更多信息，请参阅 IAM 用户指南 中的删除服务相关角色。

EC2 映像生成器 服务相关角色支持的区域

镜像生成器 支持在提供服务相关角色的所有 AWS 区域中使用该服务。有关支持的 AWS 区域列表，请参阅 AWS 区域和终端节点 (p. 5)。

对 EC2 映像生成器 身份和访问进行故障排除

可以使用以下信息，以帮助您诊断和修复在使用 镜像生成器 和 IAM 时可能遇到的常见问题。

我无权在 镜像生成器 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。

如果 `mateojackson` IAM 用户尝试使用控制台查看有关组件的详细信息，但没有 `imagebuilder:ListComponents` 权限，则会出现以下示例错误。

```
User: arn:aws-cn:iam::123456789012:user/mateojackson is not authorized to perform:
 imagebuilder:ListComponents on resource: Component
```

在这种情况下，Mateo 请求管理员更新其策略，以允许他使用 `imagebuilder:ListComponents` 操作访问 `Component` 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 EC2 Image Builder。

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 EC2 Image Builder 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-cn: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 镜像生成器

要允许其他人访问 EC2 Image Builder，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 EC2 Image Builder 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。

我希望允许我的 AWS 账户之外的人员访问我的 镜像生成器 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略的服务，您可以使用这些策略为这些人员授予您的资源的访问权限。

要了解更多信息，请参阅以下内容：

- 要了解 EC2 映像生成器 是否支持这些功能，请参阅EC2 映像生成器 如何与 IAM 一起使用 (p. 57)。
- 要了解如何为您拥有的 AWS 账户中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何为第三方 AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南 中的为第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南 中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南 中的 IAM 角色与基于资源的策略有何不同。

EC2 映像生成器 的合规性验证

EC2 映像生成器 不在任何 AWS 合规性计划范围内。

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 镜像生成器 时的合规性责任由您的数据的敏感性、贵公司的合规性目标以及适用的法律法规决定。AWS 提供以下资源以帮助满足合规性要求：

- 安全性与合规性快速入门指南安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 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 开发人员指南中的使用规则评估资源 – 此 AWS Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- 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。

要满足 STIG 合规性要求，请使用 Amazon 在 镜像生成器 服务中提供的 STIG 组件，以帮助您扫描错误的配置并运行修复脚本。我们不能保证使用 镜像生成器 生成的镜像符合 STIG 要求。您必须向合规团队确认镜像合规性。有关通过 镜像生成器 提供的 STIG 组件的完整列表，请参阅 EC2 映像生成器 STIG 组件 (p. 47)。

EC2 映像生成器 中的恢复功能

AWS 全球基础设施围绕 AWS 区域和可用区构建。AWS 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

EC2 映像生成器 服务允许您将在一个区域中生成的镜像分发到其他区域，从而为它们提供多区域 AMI 恢复功能。没有“备份”镜像管道、配方或组件的机制。您可以将配方和组件文档存储在 镜像生成器 服务外部，例如，存储在 Amazon S3 存储桶中。

无法为 EC2 映像生成器 配置高可用性 (HA)。您可以将镜像分发到多个区域以提高镜像的可用性。

EC2 映像生成器 中的基础设施安全性

作为一项托管服务，EC2 映像生成器 由 Amazon Web Services：安全流程概述白皮书中所述的 AWS 全球网络安全流程提供保护。

您可以使用 AWS 发布的 API 调用通过网络访问 镜像生成器。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。

用于通过 镜像生成器 生成镜像和运行测试的实例必须有权访问 Amazon EC2 Systems Manager 服务。因此，要使用该服务，不能禁止 Internet 访问。

EC2 映像生成器 不支持 Amazon VPC 终端节点 (PrivateLink)。

EC2 映像生成器 中的补丁管理

EC2 映像生成器 提供最新的 Amazon Linux 2 和 Windows 2012 R2 和更高版本的 AMI 以作为管理的镜像源。将定期修补这些镜像（在 镜像生成器 服务外部）。您可以根据承担 Amazon EC2 系统修补责任。如果

可以轻松替换您的应用程序工作负载中的 Amazon EC2 实例，则更新基本 AMI 并根据该镜像重新部署所有计算节点可能会更高效。

EC2 映像生成器 的安全最佳实践

EC2 映像生成器 提供了在您开发和实施自己的安全策略时需要考虑的一些安全功能。以下最佳实践是一般准则，并不代表完整的安全解决方案。由于这些最佳实践可能不适合您的环境或不满足您的环境要求，因此将其视为有用的考虑因素而不是惯例。

- 不要在 镜像生成器 配方中使用过于宽松的安全组。
- 不要与您不信任的账户共享镜像。
- 不要公开包含私有或敏感数据的镜像。
- 在镜像生成期间应用所有可用的 Windows 或 Linux 安全补丁。

脚本执行

在使用 EC2 映像生成器 生成 Linux 镜像时，AWS 将强制执行一个脚本，该脚本将在镜像生成过程结束时运行。同样，在自定义 Windows 镜像后，EC2 映像生成器 将运行 Microsoft 的 Sysprep 实用程序。这些操作遵循 AWS 强化和清理映像的最佳实践。。不过，由于可以在镜像自定义期间进行其他自定义，AWS 不能保证生成的镜像符合任何特定的法规条件。

AWS 建议您测试镜像，以验证安全状况和适用的安全合规性级别。

在使用 EC2 映像生成器 自定义 Amazon Linux 2 镜像时，以下脚本将作为必需步骤运行。

```
#!/bin/bash

FILES=(
        # Secure removal of list of sudo users
        "/etc/sudoers.d/90-cloud-init-users"

        # Secure removal of RSA encrypted SSH host keys.
        "/etc/ssh/ssh_host_rsa_key"
        "/etc/ssh/ssh_host_rsa_key.pub"

        # Secure removal of ECDSA encrypted SSH host keys.
        "/etc/ssh/ssh_host_ecdsa_key"
        "/etc/ssh/ssh_host_ecdsa_key.pub"

        # Secure removal of ED25519 encrypted SSH host keys.
        "/etc/ssh/ssh_host_ed25519_key"
        "/etc/ssh/ssh_host_ed25519_key.pub"

        # Secure removal of "root" user approved SSH keys list.
        "/root/.ssh/authorized_keys"

        # Secure removal of "ec2-user" user approved SSH keys list.
        "/home/ec2-user/.ssh/authorized_keys"

        # Secure removal of file which tracks system updates
        "/etc/.updated"
        "/var/.updated"

        # Secure removal of file with aliases for mailing lists
        "/etc/aliases.db"

        # Secure removal of file which contains the hostname of the system
        "/etc/hostname"
```

```
        # Secure removal of files with system-wide locale settings
        "/etc/locale.conf"

        # Secure removal of cached GPG signatures of yum repositories
        "/var/cache/yum/x86_64/2/.gpgkeyschecked.yum"

        # Secure removal of audit framework logs
        "/var/log/audit/audit.log"

        # Secure removal of boot logs
        "/var/log/boot.log"

        # Secure removal of kernel message logs
        "/var/log/dmesg"

        # Secure removal of cloud-init logs
        "/var/log/cloud-init.log"

        # Secure removal of cloud-init's output logs
        "/var/log/cloud-init-output.log"

        # Secure removal of cron logs
        "/var/log/cron"

        # Secure removal of aliases file for the Postfix mail transfer agent
        "/var/lib/misc/postfix.aliasesdb-stamp"

        # Secure removal of master lock for the Postfix mail transfer agent
        "/var/lib/postfix/master.lock"

        # Secure removal of spool data for the Postfix mail transfer agent
        "/var/spool/postfix/pid/master.pid"

        # Secure removal of history of Bash commands
        "/home/ec2-user/.bash_history"

        # Secure removal of file which relabels all files in the next boot
        "/.autorelabel"
)

for FILE in "${FILES[@]}"; do
        echo "Deleting $FILE"
        sudo shred -zuf $FILE
        if [[ -f $FILE ]]; then
                echo "Failed to delete '$FILE'. Failing."
                exit 1
        fi
done

# Secure removal of TOE's log directories
echo "Deleting {{workingDirectory}}/TOE_*"
sudo find {{workingDirectory}}/TOE_* -type f -exec shred -zuf {} \;
if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
fi
sudo rm -rf {{workingDirectory}}/TOE_*
if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
fi

# Secure removal of system activity reports/logs
echo "Deleting /var/log/sa/sa*"
sudo shred -zuf /var/log/sa/sa*
if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
```

```
        echo "Failed to delete /var/log/sa/sa*"
        exit 1
fi

# Secure removal of SSM logs
echo "Deleting /var/log/amazon/ssm/*"
sudo find /var/log/amazon/ssm -type f -exec shred -zuf {} \;
if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
fi
sudo rm -rf /var/log/amazon/ssm
if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
fi

# Secure removal of DHCP client leases that have been acquired
echo "Deleting /var/lib/dhclient/dhclient*.lease"
sudo shred -zuf /var/lib/dhclient/dhclient*.lease
if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt 0 ]]; then
        echo "Failed to delete /var/lib/dhclient/dhclient*.lease"
        exit 1
fi

# Secure removal of cloud-init files
echo "Deleting /var/lib/cloud/*"
sudo find /var/lib/cloud -type f -exec shred -zuf {} \;
if [[ $( sudo find /var/lib/cloud -type f | sudo wc -l ) -gt 0 ]]; then
        echo "Failed to delete /var/lib/cloud"
        exit 1
fi
sudo rm -rf /var/lib/cloud/*
if [[ $( sudo ls /var/lib/cloud | sudo wc -l ) -gt 0 ]]; then
        echo "Failed to delete /var/lib/cloud/*"
        exit 1
fi

# Secure removal of temporary files
echo "Deleting /var/tmp/*"
sudo find /var/tmp -type f -exec shred -zuf {} \;
if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/tmp"
        exit 1
fi
sudo rm -rf /var/tmp/*
if [[ $( sudo ls /var/tmp | sudo wc -l ) -gt 0 ]]; then
        echo "Failed to delete /var/tmp/*"
        exit 1
fi

# Shredding is not guaranteed to work well on rolling logs

# Removal of system logs
echo "Deleting /var/lib/rsyslog/imjournal.state"
sudo shred -zuf /var/lib/rsyslog/imjournal.state
sudo rm -f /var/lib/rsyslog/imjournal.state
if [[ -f "/var/lib/rsyslog/imjournal.state" ]]; then
        echo "Failed to delete /var/lib/rsyslog/imjournal.state"
        exit 1
fi

# Removal of journal logs
echo "Deleting /var/log/journal/*"
sudo find /var/log/journal/ -type f -exec shred -zuf {} \;
sudo rm -rf /var/log/journal/*
```

```
if [[ $( sudo ls /var/log/journal/ | sudo wc -l ) -gt 0 ]]; then
        echo "Failed to delete /var/log/journal/*"
        exit 1
fi
```

EC2 映像生成器 故障排除

EC2 映像生成器 与 AWS 服务集成在一起以进行监控和故障排除，从而帮助您解决映像生成问题。EC2 映像生成器 跟踪并显示映像生成过程中的每个步骤的进度。可以将日志导出到您提供的 Amazon S3 位置。对于高级故障排除，您可以使用 AWS Systems Manager (SSM) Run Command 运行预定义的命令和脚本。

常规故障排除

如果 镜像生成器 管道发生故障，Image Builder 将返回描述故障的错误消息。镜像生成器 还会在故障消息中返回 SSM 执行 ID，例如，以下示例中的 ID。

```
SSM execution 'aaaaaaaa-bbbb-cccc-dddd-example12345' failed with status…
```

在生成映像时，镜像生成器 使用 AWS Systems Manager (SSM) Automation 编排操作。要查看其他详细信息以帮助排除生成故障，请在 SSM Automation 控制台中搜索 镜像生成器 提供的执行 ID，然后检查 Automation 执行。

如果在您的账户中启用了 AWS CloudTrail，则会在其中记录所有生成活动。按源“ssm.amazonaws.com”筛选 CloudTrail 事件，或搜索在执行日志中返回的 Amazon EC2 实例 ID 以查看有关管道执行的更多详细信息。

默认情况下，在管道完成时，用于生成和测试活动的 Amazon EC2 实例将终止。如果遇到管道故障，您可以选择保留该实例以进行故障排除。如果遇到故障并希望访问该实例以调试问题，请为管道取消选择“Terminate instance on failure”(失败时终止实例) 选项。

您发送到 S3 存储桶的日志将显示映像生成过程中 EC2 实例上的活动的步骤和错误消息。这些日志包含了来自组件管理器的日志输出、已执行组件的定义以及在实例上执行的所有步骤的详细输出（采用 JSON 编写）。如果遇到问题，则应从 `application.log` 开始查看这些文件，以诊断实例上的问题原因。

诊断场景

- 生成失败并显示“AccessDenied: Access Denied status code: 403”(AccessDenied: 访问被拒绝状态代码: 403)
  - 问题：实例配置文件角色没有所需的权限以访问组件使用的 API 或资源，或没有所需的权限以记录到 S3。通常，在实例配置文件角色没有 S3 存储桶的 PutObject 权限时，或者在实例配置文件没有与之关联的以下角色策略时，将会发生这种情况：EC2InstanceProfileForImageBuilder 和 AmazonSSMManagedInstanceCore。
  - 解决方法：在实例配置文件角色中添加一个策略以授予访问配方中使用的 API 和资源的权限，或者将 EC2InstanceProfileForImageBuilder 和 AmazonSSMManagedInstanceCore IAM 角色策略附加到实例配置文件。
  - 问题：实例配置文件角色没有所需的权限以访问组件使用的 API 或资源，或没有所需的权限以记录到 S3。通常，在实例配置文件角色没有 S3 存储桶的 PutObject 权限时，将会发生这种情况。
  - 解决方法：在实例配置文件角色中添加一个策略以授予访问配方中使用的 API 和资源的权限，然后再次运行管道。
- 生成失败并显示“status = 'TimedOut'”(状态 =“超时”) 和“failure message = 'Step timed out while step is verifying the SSM Agent availability on the target instance(s)'”(失败消息 =“在步骤验证目标实例上的 SSM 代理可用性时，步骤超时”)
  - 问题：启动以执行生成操作和执行组件的实例无法访问 Systems Manager 终端节点。

- 解决方法：确保用于生成映像的子网有权访问并路由到 Systems Manager 终端节点。

AWS 词汇表

有关最新 AWS 术语，请参阅 AWS General Reference 中的 AWS 词汇表。