aws

用户指南

EC2 Image Builder

EC2 Image Builder: 用户指南

Table of Contents

Image Builder 是什么？

EC2 Image Builder 是一款完全托管 AWS 服务 的，可帮助您自动创建、管理和部署自定义、安全的 up-to-date服务器映像。您可以使用 AWS 管理控制台 AWS Command Line Interface、或 APIs 在中创建自定义图像 AWS 账户。

您在自己的账户中用 Image Builder 创建的自定义映像归您所有。您可以配置管道以对您拥有的映像进行自动化更新和为系统打补丁。您也可以运行独立命令，使用您定义的配置资源创建映像。

Image Builder 管道向导可以指导您通过以下步骤创建自定义映像，如下所示：

步骤 1：指定管道详细信息

- 为您的管道命名并添加标签。
- 定义元数据和漏洞扫描设置。
- 设置管道计划。

步骤 2：自定义映像

您可以选择现有配方或创建新配方。

- 为自定义选择基础映像。
- 在基础映像中添加软件或从中移除软件。
- 使用构建组件自定义设置和脚本。
- 选择要运行的测试组件。

步骤 3：定义工作流

映像工作流定义了 Image Builder 在映像创建过程的构建和测试阶段执行的步骤顺序。这是整个 Image Builder 工作流框架的一部分。

步骤 4：配置构建基础设施

- 选择要与 Image Builder 在映像创建过程中启动的实例的实例配置文件关联的 IAM 角色。
- 选择一个或多个可在启动时应用的实例类型。
- 选择 Amazon Simple Notification Service (SNS) 主题用于接收来自 Image Builder 的通知。

- 指定适用于映像创建过程的 VPC、子网和安全组。
- 选择故障排除设置，例如 Image Builder 写入日志的位置，以及在出现故障时是终止构建实例（默认）还是保持其正常运行来进行进一步的故障排除。

步骤 5：定义映像分配

- 选择 Imag AWS 区域 e Builder 分发您的亚马逊系统映像 (AMI) 或容器映像的位置。
- 如果您的 Image Builder 管道创建了 AMI，Image Builder 还支持以下配置：
  - 选择用于加密的 KMS 密钥。
  - 在 Organizations AWS 账户 之间配置 AMI 共享。
  - 将 License Manager 自管理许可证与分配的映像相关联。
  - 为您的映像配置启动模板。

Image Builder 的特征

EC2 Image Builder 提供以下功能：

提高生产力，减少建筑物合规和 up-to-date图像的操作

Image Builder 通过自动完成构建管道，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

Image Builder 提供对测试组件的访问权限，您可以在部署之前使用这些组件测试映像。您也可以使用 AWS Task Orchestrator and Executor (AWSTOE) 创建自定义测试组件，然后使用这些组件。只有在成功完成所有配置的测试时，Image Builder 才会分配您的映像。

提高部署安全性

通过使用 Image Builder，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 安全设置来创建符合行业和内部安全标准的安全out-of-the-box 映像。Image Builder 还为受监管行业的公司提供设置集合。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 Image Builder 提供的 STIG 组件的完整列表，请参阅 Image Builder 的 Amazon 托管 STIG 固化组件。

集中的实施和跟踪管理

通过使用与的内置集成 AWS Organizations，Image Builder 使您能够强制执行政策，限制账户只能在获得批准 AMIs后运行实例。

简化 AWS 账户间的资源共享

EC2 Image Builder 与 AWS Resource Access Manager (AWS RAM) 集成，允许您与任何资源 AWS 账户 或通过共享某些资源 AWS Organizations。可以共享的 EC2 Image Builder 资源有：

- 组件
- 图片
- 映像配方
- 容器配方数

有关更多信息，请参阅 与共享 Image Builder 资源 AWS RAM。

支持的操作系统

Image Builder 支持以下操作系统版本：

操作系统/发行	支持的版本
Amazon Linux	2 和 2023
CentOS	7 和 8
CentOS Stream	8
macOS	12.x（蒙特雷）、13.x（文图拉）、14.x（索诺玛）、15.x（红杉）、26.x（Tahoe）
Red Hat Enterprise Linux (RHEL)	7、8、9 和 10
SUSE Linux Enterprise Server (SUSE)	12、15 和 16
Ubuntu	18.04 LTS、20.04 LTS、22.04 LTS 和 24.04 LTS
Windows Server	2012 R2、2016、2019、2022 和 2025

支持的映像格式

对于创建亚马逊机器映像 (AMI) 的自定义映像，您可以选择现有的 AMI 作为起点。对于 Docker 容器镜像，您可以选择托管在上的公共镜像 DockerHub、Amazon ECR 中的现有容器镜像或亚马逊管理的容器镜像作为起点。

默认限额

要查看 Image Builder 的默认配额，请参阅 Image Builder 终端节点和配额。

AWS 区域和终端节点

要查看 Image Builder 的服务终端节点，请参阅 Image Builder 终端节点和配额。

概念

以下术语和概念对您了解和使用 EC2 Image Builder 非常有用。

AMI

亚马逊机器映像（AMI）是 Amazon EC2 中的基本部署单元，也是您可以使用 Image Builder 创建的映像类型之一。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 (OS) 和预装软件。有关更多信息，请参阅 亚马逊机器映像 (AMI)。

映像管道

图像管道提供了一个自动化框架，用于在上构建 AMIs 安全的容器映像 AWS。Image Builder 映像管道与一个映像配方或容器配方相关联，该配方定义映像构建生命周期的构建、验证和测试阶段。

映像管道可与定义映像构建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。还可以将映像管道与分发配置相关联，以定义您希望如何部署映像。

托管映像

托管映像是 Image Builder 中的一种资源，由 AMI 或容器映像以及版本和平台等元数据组成。Image Builder 管道使用托管映像来确定构建时使用哪个基础映像。在本指南中，托管映像有时称为“映像”，但是，映像与 AMI 不同。

映像配方

Image Builder 映像配方是一个文档，用于定义基础映像和要应用于基础映像以生成输出 AMI 映像所需配置的组件。您可以使用映像配方复制构建。可以使用控制台向导、或 API 共享、分支和编辑 Image Builder 图像配方。 AWS CLI您可以将映像配方与版本控制软件一起使用，以维护可共享的版本化映像配方。

容器配方

Image Builder 映像配方是一个文档，用于定义基础映像和要应用于基础映像以生成输出容器映像所需配置的组件。您可以使用容器配方复制构建。可以使用控制台向导、 AWS CLI或 API 共享、分支和编辑 Image Builder 映像配方。您可以将容器配方与版本控制软件一起使用，以维护可共享的版本化容器配方。

基础映像

基础映像是映像或容器配方文档中与组件一起使用的选定映像和操作系统。基础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自定义实例（构建组件）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。

组件是根据声明式纯文本 YAML 或 JSON 文档创建的，该文档描述用于构建、验证或测试由管道生成的实例的运行时配置。组件使用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运行所需的步骤。

创建组件后，使用映像配方或容器配方将它们分成一个或多个一组，用于定义构建和测试虚拟机或容器映像的计划。您可以使用由其拥有和管理的公共组件 AWS，也可以创建自己的组件。有关组件的更多信息，请参阅 Image Builder 如何使用 AWS Task Orchestrator and Executor 应用程序管理组件。

组件文档

这是一个声明性的纯文本 YAML 或 JSON 文档，描述可应用于映像的自定义配置。该文档用于创建构建或测试组件。

运行时阶段

EC2 Image Builder 有两个运行时阶段：构建和测试。每个运行时阶段都有一个或多个阶段，其配置由组件文档定义。

配置阶段

以下列表显示了在构建和测试阶段运行的阶段：

构建阶段：

构建阶段

映像管道在运行时从构建阶段的构建时段开始。下载基础映像，并应用为组件的构建阶段指定的配置,以构建和启动实例。

验证阶段

在 Image Builder 启动实例并应用所有构建阶段自定义后，验证阶段开始。在此阶段，根据组件为验证阶段指定的配置，Image Builder 确保所有自定义项均按预期运行。如果实例验证成功，Image Builder 将停止实例，创建映像，然后继续测试阶段。

测试阶段：

测试时段

在此阶段，Image Builder 会根据验证阶段成功完成后创建的映像启动实例。Image Builder 在此阶段运行测试组件，以验证实例是否正常并按预期运行。

容器主机测试阶段

在 Image Builder 对您在容器配方中选择的所有组件运行测试阶段后，Image Builder 会针对容器工作流程运行此阶段。容器主机测试阶段可以运行其他测试，以验证容器管理和自定义的运行时配置。

工作流

工作流程定义了 Image Builder 在创建新映像时执行的步骤顺序。所有映像都有构建、测试和分发工作流程。

工作流程类型

`BUILD`

涵盖所创建的每个映像的构建阶段配置。

`TEST`

涵盖所创建的每个映像的测试阶段配置。

`DISTRIBUTION`

涵盖创建的每个映像的分发阶段配置。

定价

使用 EC2 Image Builder 创建自定义 AMI 或容器映像，无需任何费用。但是，标准定价仍适用于该过程中使用的其他服务。以下列表包括在创建、构建、存储和分发自定义 AMI 或容器映像时可能会产生费用的某些 AWS 服务 镜像的使用情况，具体取决于您的配置。

- 启动 EC2 实例
- 在 Amazon S3 上存储日志
- 使用 Amazon Inspector 验证映像
- 为您存储 Amazon EBS 快照 AMIs
- 将容器映像存储在 Amazon ECR 中
- 将容器映像推入和拉出 Amazon ECR
- 如果启用 Systems Manager 高级套餐并运行具有本地激活的 Amazon 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 Image Builder 使用其他 AWS 服务 来构建映像，具体取决于您的映像生成器配方配置。有关自定义映像的产品和服务集成的更多信息，请参阅将产品和服务集成到 Image Builder 中。

EC2 Image Builder 的工作原理

当您使用 EC2 Image Builder 控制台创建自定义图像管道时，系统会指导您完成以下步骤。

1. 指定管道详细信息-输入有关您管道的信息，例如名称、描述、标签和运行自动构建的计划。您也可以选择手动构建。
2. 选择配方-在构建 AMI 或构建容器映像之间进行选择。对于这两种类型的输出映像，您可以输入配方的名称和版本，选择基础映像，然后选择要添加的组件以进行构建和测试。您也可以选择自动版本控制，以确保您的基础映像始终使用最新的可用操作系统 (OS) 版本。容器配方还定义了 Dockerfiles 以及输出 Docker 容器映像的目标 Amazon ECR 存储库。

> **Note**
>
> 组件是映像配方或容器配方所消耗的构建基块。例如，安装软件包、安全强化步骤和测试。选定的基本映像和组件组成了镜像配方。

3. 定义基础架构配置 — Image Builder 在您的账户中启动 EC2 实例，以自定义图像并运行验证测试。基础设施配置设置为在构建 AWS 账户 过程中将在您的中运行的实例指定基础设施详细信息。
4. 定义分配设置-在构建完成并通过所有测试后，选择要将映像分配到的 AWS 区域。管道会自动将您的映像分配到它运行构建的区域，并且您可以为其他区域添加映像分配。

您根据自定义基础映像构建的映像位于您的 AWS 账户。您可以输入生成计划，以便配置镜像管道以生成更新和修补的映像版本。在生成完成后，您可以通过 Amazon Simple Notification Service (SNS) 接收通知。除了生成最终映像外，Image Builder 控制台向导还会生成一个可用于现有版本控制系统和持续 integration/continuous 部署 (CI/CD) 管道的配方，以实现可重复的自动化。您可以共享和创建新的配方版本。

本节内容

AMI 元素

Amazon 系统映像 (AMI) 是一种预配置的虚拟机 (VM) 映像，其中包含用于部署 EC2 实例的操作系统和软件。

AMI 包括以下元素：

- 虚拟机根卷的模板。当您启动 Amazon EC2 虚拟机时，根设备卷包含用于启动实例的映像。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 根设备音量。
- 在使用时，根设备是通过 EBS 快照创建的 EBS 卷。
- 启动权限决定了 AWS 账户 可以通过 AMI 启动 VMs 的权限。
- 块储存设备映射数据，用于指定在启动后附加到实例的卷。
- 每个区域、每个账户的唯一资源标识符。
- 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）。

- Windows 映像的 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。

组件管理

EC2 Image Builder 使用组件管理应用程序 AWS Task Orchestrator and Executor (AWSTOE)，它可以帮助您协调复杂的工作流程、修改系统配置以及使用基于 YAML 的脚本组件测试系统。由于它 AWSTOE 是一个独立的应用程序，因此不需要任何其他设置。它可以在任何云基础设施和本地运行。要开始 AWSTOE 将其用作独立应用程序，请参阅手动设置以开发自定义组件 AWSTOE。

Image Builder 用于 AWSTOE 执行所有实例上的活动。其中包括在拍摄快照之前构建和验证映像，以及在创建最终映像之前测试快照以确保其按预期运行。有关 Image Builder AWSTOE 如何使用管理其组件的更多信息，请参阅使用组件自定义 Image Builder 映像。有关创建组件与 AWSTOE的详细信息，请参阅 Image Builder 如何使用 AWS Task Orchestrator and Executor 应用程序管理组件。

映像测试

在创建最终映像之前，您可以使用 AWSTOE 测试组件来验证映像，并确保其按预期运行。

通常，每个测试组件由一个 YAML 文档构成，此文档包含测试脚本、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）。

创建的资源

创建管道时，除非满足以下条件，否则不会创建 Image Builder 外部的资源：

- 通过管道计划创建映像时
- 当您从 Image Builder 控制台的操作菜单中选择运行管道时
- 当你从 API 或者 AWS CLI:StartImagePipelineExecution或运行以下任一命令时 CreateImage

以下资源是在映像构建过程中创建的：

AMI 镜像管道

- EC2 实例（临时）
- EC2 实例上的 System EnhancedImageMetadata s Manager 库存关联（如果已启用，则通过 Systems Manager 状态管理器）

- 亚马逊 EC2 AMI
- 与亚马逊 AMI 相关的亚马逊 EBS 快照 EC2

容器映像管道

- 在 EC2 实例上运行的 Docker 容器（临时）
- EC2 实例上的 Systems Manager 库存关联（通过 System `EnhancedImageMetadata` s Manager 状态管理器）已启用）
- Docker 容器映像
- Dockerfile

创建映像后，所有临时资源都将被删除。

分配

EC2 Image Builder 可以将映像分发 AMIs 或容器到任何 AWS 区域。映像将被复制到您在用于生成镜像的账户中指定的每个区域。

对于 AMI 输出映像，您可以定义 AMI 启动权限以控制 AWS 账户 哪些允许使用已创建的 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。如果您既将 AMI 分发到其他区域，又为其他账户定义启动权限，则启动权限将传播到分发 AMI 的所有区域。 AMIs

您还可以使用您的 AWS Organizations 账户对成员账户实施限制，使其只能在获得批准且合规的情况下启动实例 AMIs。有关更多信息，请参阅在组织 AWS 账户 中管理。

要使用 Image Builder 控制台更新您的分配设置，请按照 通过控制台创建新的映像配方版本 或 使用控制台创建新的容器配方版本 的步骤操作。

共享资源

要与其他账户或其内部共享组件、配方或图像 AWS Organizations，请参阅与共享 Image Builder 资源 AWS RAM。

合规

对于互联网安全中心 (CIS) 基准测试，I EC2 mage Builder 使用 Amazon Inspector 对风险敞口、漏洞以及与最佳实践和合规标准的偏差进行评估。例如，Image Builder 会评估非预期的网络可访问性、未修补的网络连接 CVEs、公共互联网连接和远程根登录激活。Amazon Inspector 作为测试组件提供，

您可以选择将其添加到您的映像配方中。有关 Amazon Inspector 的更多信息，请参阅亚马逊 Inspector 用户指南。有关更多信息，请参阅 Center for Internet Security (CIS) Benchmarks。

Image Builder 提供 STIG 强化组件，可帮助您更有效地构建符合基准 STIG 标准的兼容映像。这些 STIG 组件扫描错误的配置并运行修复脚本。使用符合 STIG 要求的组件不会收取额外的费用。有关通过 Image Builder 提供的 STIG 组件的完整列表，请参阅 Image Builder 的 Amazon 托管 STIG 固化组件。

Image Builder 中的语义版本控制

Image Builder 使用语义版本控制来组织资源并确保资源具有唯一性。 IDs语义版本有四个节点：

<major>。 *<minor>*。 *<patch>*/<build>

您可以为前三个分配值，并且可以筛选所有这些值。

语义版本控制包含在每个对象的 Amazon 资源名称 (ARN) 中，其级别适用于该对象，如下所示：

1. 无版本 ARNs 和 Name 在任何节点中都 ARNs 不包含特定值。节点要么完全省略，要么被指定为通配符，例如：x.x.x。
2. 版本 ARNs 只有前三个节点:<major>. <minor>。 <patch>
3. 构建版本 ARNs 包含所有四个节点，并指向对象的特定版本的特定构建。

分配：对于前三个节点，您可以为每个节点分配任何正整数值或零，上限为 2^30-1 或 1073741823。映像生成器会自动将内部版本号分配给第四个节点。

模式：您可以使用符合可分配节点分配要求的任何数字模式。例如，您可以选择软件版本模式（例如 1.0.0）或日期（例如 2021.01.01）。

选择：通过语义版本控制，您可以灵活地使用通配符（x）在为食谱选择基本映像或组件时指定最新版本或节点。在任何节点中使用通配符时，第一个通配符右侧的所有节点必须也是通配符。

例如，给定以下最新版本：2.2.4、1.7.8 和 1.6.8，使用通配符选择版本会产生以下结果：

- `x.x.x` = 2.2.4
- `1.x.x`= 1.7.8
- `1.6.x`= 1.6.8
- `x.2.x` 无效，并产生了错误

- `1.x.8` 无效，并产生了错误

生命周期策略中的通配符版本控制

在生命周期策略配方选择中，您可以对语义版本使用通配符模式，以使用单个策略定位配方的多个版本。这样就无需为每个配方版本创建单独的策略，从而简化了生命周期管理。

生命周期策略配方版本支持以下通配符模式：

- `x.x.x`— 匹配食谱的所有版本。
- `1.x.x`— 匹配主版本 1 中的所有次要版本和补丁版本。
- `1.0.x`— 匹配版本 1.0 中的所有补丁版本。

运行带有通配符模式的生命周期策略时，Image Builder 会在执行时将通配符解析为所有匹配的配方版本。这将为该执行创建不可变的版本列表。在策略执行开始后创建的新配方版本将自动包含在下一次计划执行中。

有关使用通配符版本控制创建生命周期策略的更多信息，请参阅。创建生命周期策略

使用版本引用

版本引用是 ready-to-use ARN 字符串，它根据您创建或检索的资源的语义版本包含通配符模式。Image Builder 不是编写自定义代码来解析 ARNs 和插入通配符，而是为你完成这项工作。

当您创建或检索 Image Builder 资源时，Image Builder 会自动在对象中提供 ARNs 带有通配符的`latestVersionReferences`预构版本模式。 ARNs 当你想使用通配符版本控制模式引用资源时，这样就无需手动解析和重建。

例如，当您创建带有版本`1.2.3`的组件时，Image Builder 会返回：

```
{
    "componentBuildVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
my-component/1.2.3/1",
    "latestVersionReferences": {
        "latestVersionArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-
component/x.x.x",
        "latestMajorVersionArn": "arn:aws:imagebuilder:us-
west-2:123456789012:component/my-component/1.x.x",
```

```
        "latestMinorVersionArn": "arn:aws:imagebuilder:us-
west-2:123456789012:component/my-component/1.2.x",
        "latestPatchVersionArn": "arn:aws:imagebuilder:us-
west-2:123456789012:component/my-component/1.2.3"
    }
}
```

可用的版本参考模式

该latestVersionReferences对象包含四个 ARN 模式：

- latestVersionArn (x.x.x)-始终解析为绝对最新版本。
- atestMajorVersionArn (1.x.x)-解析为主要版本中的最新次要版本和补丁版本。
- latestMinorVersionArn (1.2.x)-解析为特定次要版本中的最新补丁版本。
- latestPatchVersionArn (1.2.3)-引用特定的语义版本并解析为支持多个编译版本的资源的最新编译版本。

返回版本引用的资源

所有版本控制的 Image Builder 资源均返回版本引用：CreateGet APIs

- 组件-CreateComponent，GetComponent
- 图片食谱-CreateImageRecipe，GetImageRecipe
- 容器食谱-CreateContainerRecipe, GetContainerRecipe
- 图片-CreateImage，GetImage
- 工作流程-CreateWorkflow，GetWorkflow

注意：由于Image Builder要求您始终使用最新版本的图像生成器管理的工作流程，因此仅latestVersionArn (x.x.x)返回图像生成器管理的工作流程。

准备好使用 Image Builder 构建自定义映像

使用 EC2 Image Builder 创建映像之前，请验证您是否满足创建映像管道的以下先决条件。除非另有说明，否则所有类型的管道都需要满足这些先决条件。

先决条件

- Image Builder 服务相关角色
- 配置要求
- 适用于容器映像管道的容器存储库
- 保存 macOS 映像的专属主机
- IAM 先决条件
- Systems Manager Agent 先决条件

满足先决条件后，您可以通过以下任一界面管理 EC2 Image Builder。

- EC2 Image Builder 控制台
- 中的 Image Builder 命令 AWS CLI
- EC2 Image Builder API 参考
- AWS SDKs 和工具

Image Builder 服务相关角色

EC2 Image Builder 使用服务相关角色代表您向其他 AWS 服务授予权限。您无需手动创建服务关联角色。当您在 AWS 管理控制台、或 AWS API 中创建第一个 Image Builder 资源时，Image Builder 会为您创建服务相关角色。 AWS CLI有关 Image Builder 在您的账户中创建服务相关角色的更多信息，请参阅 使用 Image Builder 的 IAM 服务相关角色。

配置要求

- Image Builder 支持 AWS PrivateLink。有关为 Image Builder 配置 VPC 终端节点的更多信息，请参阅 Image Builder 和 AWS PrivateLink 接口 VPC 终端节点。
- Image Builder 用于构建容器映像的实例必须具有互联网访问权限才能 AWS CLI 从 Amazon S3 下载容器映像，并从 Docker Hub 存储库下载基础映像（如果适用）。Image Builder 使用从容器配方中获取 Dockerfile，该文件作为数据存储。 AWS CLI

- Image Builder 用于构建映像和运行测试的实例必须有权访问 Systems Manager 服务。安装要求取决于您的操作系统。

  要查看基础映像的安装要求，请选择与基础映像操作系统相匹配的选项卡。

  Linux

  对于 Amazon EC2 Linux 实例，如果尚未安装 Systems Manager 代理，Image Builder 会在构建实例上安装该代理，并在创建映像前将其移除。

  Windows

  Image Builder 不会在 Amazon EC2 Windows Server 构建实例上安装 Systems Manager 代理。如果基础映像未预装 Systems Manager 代理，则必须从源映像启动实例，在实例上手动安装 Systems Manager，然后从实例创建新的基础映像。

  要在 Amazon EC2 Windows 服务器实例上手动安装 Systems Manager 代理，请参阅 AWS Systems Manager 用户指南中的在 Windows 服务器的 EC2 实例上手动安装 Systems Manager 代理。

适用于容器映像管道的容器存储库

对于容器映像管道，配方定义了生成并存储在目标容器存储库中的 Docker 映像的配置。在为 Docker 映像创建容器配方之前，您必须先创建目标存储库。

Image Builder 使用 Amazon ECR 作为其容器映像的目标存储库。要创建 Amazon ECR 存储库，请按照 Amazon Elastic 容器注册表用户指南中的创建存储库所述步骤进行操作。

保存 macOS 映像的专属主机

Amazon EC2 Mac 实例需要一个运行于裸机实例类型的专属主机。在创建自定义 macOS 映像之前，必须为您的账户分配专属主机。有关 Mac 实例的更多信息以及原生支持 macOS 操作系统的实例类型列表，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 Mac 实例。

创建专属主机后，可以在基础设施配置资源中为映像配置设置。基础设施配置包括放置属性，您可以在其中指定从映像启动的实例应保存到的主机、主机置放群组或可用区。

IAM 先决条件

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
- EC2InstanceProfileForImageBuilderECRContainerBuilds
- Amazon SSMManaged InstanceCore

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。例如：

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "UploadFileToS3Bucket",
   "Effect": "Allow",
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::bucket-name/*"
  }
 ]
}
```

附加策略

以下步骤将指导您完成将 IAM 策略附加到 IAM 角色以授予上述权限的过程。

1. 登录 AWS 管理控制台并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 使用 EC2InstanceProfileForImageBuilder 筛选策略列表
4. 选择策略旁边的项目符号，然后在策略操作下拉列表中选择附加。
5. 选择要附加该策略的 IAM 角色的名称。
6. 选择 Attach policy（附加策略）。

7. 对EC2InstanceProfileForImageBuilderECRContainerBuilds和 Amazon SSMManaged InstanceCore 政策重复步骤 3-6。

> **Note**
>
> 如果要将使用 Image Builder 创建的映像复制到另一个账户，则必须在所有目标账户中创建 `EC2ImageBuilderDistributionCrossAccountRole` 角色，并将 Ec2ImageBuilderCrossAccountDistributionAccess 策略 托管策略附加到该角色。有关更多信息，请参阅 与共享 Image Builder 资源 AWS RAM。

Systems Manager Agent 先决条件

EC2 Image Builder 在其启动的 EC2 实例上运行 AWS Systems Manager （ Systems Manager ）代理，以构建和测试映像。Image Builder 使用 Systems Manager 清单，以收集在构建阶段使用的实例的其他信息。该信息包括操作系统 (OS) 名称和版本，以及操作系统报告的软件包及其相应版本的列表。

要选择不收集此信息，请选择与您的首选环境相匹配的方法：

- Image Builder 控制台 – 取消选择启用增强的元数据收集复选框。
- AWS CLI – 指定 `--no-enhanced-image-metadata-enabled` 选项
- I@@ mage Builder API 或 SDKs — 将`enhancedImageMetadataEnabled`参数设置为`false`。

Image Builder 使用 `RunCommand` 将操作发送到生成和测试实例，以作为映像构建和测试工作流程的一部分。您不能选择停止使用 `RunCommand` 将操作发送到生成和测试实例。

通过 Image Builder 教程学习如何创建自定义映像

使用 EC2 Image Builder 构建自定义映像和组件的方法有很多。教程可帮助您了解 Image Builder 的关键概念。每个教程都提供了一个使用案例，其中包含您首次操作可以遵循的步骤。这些说明尽可能使用默认值来协助了解整个过程。使用其中一个教程后，您可以探索更多的方法来自定义自己的映像。

构建您的第一个映像

以下教程向您展示了如何使用 Image Builder 控制台向导构建您的第一个映像。在本教程结束时，您将创建以下一组 Image Builder 资源。本教程的最后一步是清理您创建的资源。

- 亚马逊机器映像 (AMI) 的映像配方或容器映像的容器配方。
- 具有默认设置的基础设施配置资源。
- 一种具有默认设置的分配设置资源，用于将输出分配到源区域（您在区域中用于运行控制台向导的账户）。
- 一种映像管道，它使用列出的资源通过默认的映像构建工作流来构建输出映像。
- 输出 AMI 或容器映像。

控制台向导教程

- 管道向导：创建 AMI
- 管道向导：创建容器镜像

创建带有输入参数的自定义组件

以下教程向您展示了如何创建用于定义输入参数的自定义组件，然后根据您的 Image Builder 配方设置值。

创建带有参数的自定义组件

在 Image Builder 中使用 Systems Manager 参数

以下教程向您展示如何创建 P AWS Systems Manager arameter Store 参数并在图像配方中使用该参数。

在食谱中使用基本图像参数

您也可以在 AMI 分发设置中使用参数存储参数来存储您的输出图像 ID，也可以使用自定义组件中的参数存储参数。有关更多信息，创建和更新 AMI 分配配置请参见发行版和使用 Systems Manager 参数存储参数自定义组件。

教程：通过 Image Builder 控制台向导使用输出 AMI 创建映像管道

本教程将引导您使用创建映像管道控制台向导创建自动管道，以构建和维护自定义 EC2 Image Builder 映像。为了帮助您高效地完成这些步骤，系统会在默认设置可用时使用默认设置，并跳过可选部分。

创建映像管道工作流

- 步骤 1：指定管道详细信息
- 步骤 2：选择配方
- 步骤 3：定义基础设施配置-可选
- 步骤 4：定义分配设置-可选
- 步骤 5: 审核
- 步骤 6：清除

步骤 1：指定管道详细信息

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 要开始创建管道，请选择创建映像管道。
3. 在常规部分中，输入您的管道名称（必需）。

   > **Tip**
   >
   > 增强型元数据收集默认开启。为确保组件和基础映像之间的兼容性，请将其保持开启状态。

4. 在生成计划部分，您可以保留 计划选项的默认值。请注意，默认计划中显示的时区是通用协调时间 (UTC)。有关 UTC 时间的更多信息以及要查找您所在时区的偏移量，请参阅时区缩写-全球列表。

   对于依赖项更新设置，请选择Run pipeline at the scheduled time if there are dependency updates选项。此设置会让您的管道在开始构建之前检查更新。如果没有更新，它将跳过计划的管道构建。

> **Note**
>
> 为确保您的管道能够按预期识别依赖项更新和构建，您必须对基础映像和组件使用语义版本控制 (x.x.x)。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

5. 要继续执行下一个步骤，请选择下一步。

步骤 2：选择配方

1. Image Builder 在配方一节中默认为使用现有配方。首次使用时，请选择创建新配方选项。
2. 在映像类型一节，选择亚马逊机器映像 (AMI)选项以创建生成和分配 AMI 的映像管道。
3. 在常规一节，输入以下必填框：
   - 姓名—您的配方名称
   - 版本-您的配方版本（使用格式<major>.<minor>.<patch>，其中 major、minor 和 patch 是整数值）。新配方通常以`1.0.0`开头。
4. 在源映像一节中，保留选择映像、映像操作系统 (OS) 和映像来源的默认值。这将生成由亚马逊管理 AMIs 的 Linux 列表。在本教程中，选择`Amazon Linux 2 x86`图片。
   a. 从映像名称下拉列表中，选择一个映像。
   b. 保留自动版本控制选项的默认值（使用最新的可用操作系统版本）。

   > **Note**
   >
   > 此设置可确保您的管道对基础映像使用语义版本控制，以检测自动计划作业的依赖项更新。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

5. 在实例配置一节，保留 Systems Manager 代理的默认值。这会使得 Image Builder 在构建和测试完成后保留 Systems Manager 代理，以便在新映像中包含 Systems Manager 代理。

   在本教程中，将用户数据留空。当您启动构建实例时，有时可以使用此区域以提供要运行的命令或命令脚本。但是，它会替换 Image Builder 可能添加的任何命令，以确保安装 Systems Manager。使用它时，请确保您的基础映像上已预先安装了 Systems Manager 代理，或者已将安装的内容包含在用户数据中。

6. 在 “组件” 部分中，您可以选择不添加任何组件并继续。如果要添加组件，请在 “生成组件” 面板中选择 “添加构建组件”，然后Amazon managed从组件所有者筛选器列表中进行选择。这将在控制台界面的右侧打开一个选择面板，您可以在其中浏览和筛选可用的组件。

   在本教程中，请选择一个使用最新安全更新来更新 Linux 的组件，如下所示：

   a. 通过在面板顶部的搜索栏中输入单词 update 来筛选结果。

   b. 选中 update-linux 构建组件的复选框。

   c. 保留版本控制选项的默认值（使用可用的最新版本）。

      > **Note**
      >
      > 此设置可确保您的管道对选定组件使用语义版本控制，以检测自动计划作业的依赖项更新。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

   d. 选择 “添加到食谱”，将该组件添加到您的食谱中。这将关闭组件选择面板。

   e. 返回到 “生成组件” 面板中，将显示您添加的组件。

7. 重新排序组件（可选）

   如果您选择了多个组件要包含在映像中，则可以使用 drag-and-drop操作将它们重新排列为它们在构建过程中应遵循的运行顺序。

   > **Note**
   >
   > CIS 加固组件未遵循 Image Builder 配方中的标准组件排序规则。CIS 加固组件始终最后运行，以确保基准测试针对您的输出映像运行。

   a. 重复前面的步骤，将该update-linux-kernel-5组件添加到您的食谱中。

   b. 您刚刚添加的组件具有内核版本的输入参数。要展开版本控制选项或输入参数的设置，可以选择设置名称旁边的箭头。要展开所有选定组件的所有设置，可以关闭和打开全部展开开关。有关在组件中使用输入参数以及在配方中设置输入参数的更多信息，请参阅 教程：创建带有输入参数的自定义组件。

   c. 选择其中一个组件，然后向上或向下拖动以更改组件的运行顺序。

   d. 要删除 update-linux-kernel-5 组件，请从组件框的右上角选择 X。

重复此步骤以删除您可能已添加的任何其他组件，只选中该update-linux组件。

8. 要继续执行下一个步骤，请选择下一步。

步骤 3：定义基础设施配置-可选

Image Builder 在您的账户中启动 EC2 实例，以自定义映像和运行验证测试。基础设施配置设置为在构建 AWS 账户 过程中将在您的中运行的实例指定基础设施详细信息。

在基础设施配置一节中，配置选项默认为 `Create infrastructure configuration using service defaults`。这将创建一个 IAM 角色和关联的实例配置文件，供 EC2 构建与测试实例用来配置您的映像。有关基础设施配置设置的更多信息，请参阅 EC2 Image Builder API 参考CreateInfrastructureConfiguration中的。

在本教程中，我们将使用默认设置。

> **Note**
>
> 要指定用于私有 VPC 的子网，您可以创建自己的自定义基础设施配置，也可以使用已经创建的设置。

- 要继续执行下一个步骤，请选择下一步。

步骤 4：定义分配设置-可选

分发配置包括输出 AMI 名称、用于加密的特定区域设置、启动权限以及可以启动输出 AMI 的组织和组织单位 (OUs) 以及许可证配置。 AWS 账户

在分配设置一节中，配置选项默认为 `Create distribution settings using service defaults`。此选项会将输出 AMI 分配到当前区域。有关配置分配设置的更多信息，请参阅管理 Image Builder 分配设置。

在本教程中，我们将使用默认设置。

- 要继续执行下一个步骤，请选择下一步。

步骤 5: 审核

审核一节显示您配置的所有设置。要编辑任何给定一节中的信息，请选择位于步骤一节中右上角的编辑按钮。例如，如果要更改管道名称，请选择步骤 1：管道详细信息一节中右上角的编辑按钮。

1. 查看设置后，选择创建管道来创建您的管道。
2. 如您的资源是为分配设置、基础架构配置、新配方和管道而创建，您可以在页面顶部看到成功或失败的消息。要查看资源的详细信息，包括资源标识符，请选择查看详细信息。
3. 查看资源的详细信息后，您可以通过从导航窗格中选择资源类型来查看有关其他资源的详细信息。例如，要查看新管道的详细信息，请从导航窗格中选择映像管道。如果构建成功，则您的新管道将显示在映像管道列表中。

步骤 6：清除

Image Builder 环境就像家一样，需要定期维护，以帮助您找到所需的内容，并在不费吹灰之力的情况下完成任务。请务必定期对为了测试而创建的临时资源进行清理。否则，您可能会忘记这些资源，然后再也不记得它们的用途。届时，可能还不清楚您能否安全地删除它们。

> **Tip**
>
> 为防止在删除资源时出现依赖项错误，请确保按以下顺序删除资源：
>
> 1. 映像管道
> 2. 映像配方
> 3. 所有剩余的资源

要清除您为本教程创建的资源，请执行以下步骤：

删除管道

1. 要查看在您的账户下创建的构建管道列表，请从导航窗格中选择映像管道。
2. 导航到 镜像管道 页面，然后选中要删除的管道名称旁边的复选框。
3. 在映像管道面板顶部的操作菜单中，选择删除。
4. 若要确认删除，请在框中输入 `Delete`，然后选择删除。

删除配方

1. 要查看在您的账户下创建的配方列表，请从导航窗格中选择映像配方。
2. 选择配方名称旁边的复选框以选择想要删除的配方。
3. 在映像配方面板顶部的操作菜单中，选择删除配方。
4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除基础设施配置

1. 要查看在您的账户下创建的基础设施配置列表，请从导航窗格中选择基础设施配置。
2. 选择配置名称旁边的复选框以选择想要删除的基础架构配置。
3. 在基础设施配置面板的顶部，选择删除。
4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除分配设置

1. 要查看在您的账户下创建的分配设置列表，请从导航窗格中选择分配设置。
2. 选择配置名称旁边的复选框以选择您为本教程创建的分配设置。
3. 在分配设置面板的顶部，选择删除。
4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除映像

请按照以下步骤验证您是否已删除从教程管道中创建的所有映像。本教程不太可能创建映像，除非根据构建计划，自您创建管道以来已经过了足够的时间，使得它运行。

1. 要查看在您的账户下创建的映像列表，请从导航窗格中选择映像。
2. 对于要移除的映像，选择映像版本。此时将打开映像构建版本页面。
3. 选中要删除的任何图像的版本旁边的复选框。您一次可以选择多个映像版本。
4. 在映像构建版本面板的顶部，选择删除版本。
5. 若要确认删除，请在框中输入 Delete，然后选择删除。

教程：通过 Image Builder 控制台向导使用输出 Docker 容器映像创建映像管道

本教程将引导您使用创建映像管道控制台向导创建自动管道，以构建和维护自定义 EC2 Image Builder Docker 映像。为了帮助您高效地完成这些步骤，系统会在默认设置可用时使用默认设置，并跳过可选部分。

创建映像管道工作流

步骤 1：指定管道详细信息

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 要开始创建管道，请选择创建映像管道。
3. 在常规部分中，输入您的管道名称（必需）。
4. 在生成计划部分，您可以保留 计划选项的默认值。请注意，默认计划中显示的时区是通用协调时间 (UTC)。有关 UTC 时间的更多信息以及要查找您所在时区的偏移量，请参阅时区缩写-全球列表。

   对于依赖项更新设置，请选择Run pipeline at the scheduled time if there are dependency updates选项。此设置会让您的管道在开始构建之前检查更新。如果没有更新，它将跳过计划的管道构建。

   > **Note**
   >
   > 为确保您的管道能够按预期识别依赖项更新和构建，您必须对基础映像和组件使用语义版本控制 (x.x.x)。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

5. 要继续执行下一个步骤，请选择下一步。

步骤 2：选择配方

1. Image Builder 在配方一节中默认为使用现有配方。首次使用时，请选择创建新配方选项。
2. 在映像类型一节，选择Docker 映像选项以创建一个容器管道，该管道将生成 Docker 映像并将其分配到目标区域的 Amazon ECR 存储库。
3. 在常规一节，输入以下必填框：

   - 姓名—您的配方名称
   - 版本-您的配方版本（使用格式<major>.<minor>.<patch>，其中 major、minor 和 patch 是整数值）。新配方通常以`1.0.0`开头。
4. 在源映像一节中，保留选择映像、映像操作系统 (OS) 和映像来源的默认值。这会生成一个由 Amazon 管理的 Amazon Linux 2 容器映像列表，供您从中进行基础映像选择。

   a. 从映像名称下拉列表中，选择一个映像。

   b. 保留自动版本控制选项的默认值（使用最新的可用操作系统版本）。

   > **Note**
   >
   > 此设置可确保您的管道对基础映像使用语义版本控制，以检测自动计划作业的依赖项更新。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。
5. 在 “组件” 部分中，您可以选择不添加任何组件并继续。如果要添加组件，可以在 “构建组件 — Amazon Linux” 面板中浏览页面上列出的组件。使用右上角的分页控件浏览适用于基础映像操作系统的其他组件。您也可以搜索特定的组件，或者使用组件管理器创建自己的构建组件。

   在本教程中，请选择一个使用最新安全更新来更新 Linux 的组件，如下所示：

   a. 通过在面板顶部的搜索栏中输入单词 `update` 来筛选结果。

   b. 选中 `update-linux` 构建组件的复选框。

   c. 向下滚动，在选定组件列表的右上角，选择全部展开。

   d. 保留版本控制选项的默认值（使用最新的可用组件版本）。

> **Note**
>
> 此设置可确保您的管道对选定组件使用语义版本控制，以检测自动计划作业的依赖项更新。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

如果您选择了具有输入参数的组件，则还可以在此区域看到这些参数。本教程中未涵盖参数。有关在组件中使用输入参数以及在配方中设置输入参数的更多信息，请参阅 教程：创建带有输入参数的自定义组件。

重新排序组件（可选）

如果您选择了多个组件要包含在映像中，则可以使用 drag-and-drop操作将它们重新排列为它们在构建过程中应遵循的运行顺序。

> **Note**
>
> CIS 加固组件未遵循 Image Builder 配方中的标准组件排序规则。CIS 加固组件始终最后运行，以确保基准测试针对您的输出映像运行。

1. 向上滚动至可用组件列表。
2. 选中 `update-linux-kernel-mainline` 构建组件（或您选择的任何其他组件）的复选框。
3. 向下滚动到选定组件列表，查看到至少有两个结果。
4. 新添加的组件可能不会展开其版本控制。要展开版本控制选项，可以选择版本控制选项旁边的箭头，也可以关闭和打开全部展开开关以展开所有选定组件的版本控制。
5. 选择其中一个组件，然后向上或向下拖动以更改组件的运行顺序。
6. 要删除 `update-linux-kernel-mainline` 组件，请从组件框的右上角选择 X。
7. 重复上一步删除您可能已添加的任何其他组件，只选中 `update-linux` 组件。

6. 在 Dockerfile 模板一节，选择使用示例选项。模板数据由上下内容文变量组成，Image Builder 根据容器映像食谱放置构建信息或脚本。

   默认情况下，Image Builder 在 Dockerfile 中使用以下上下文变量。

parentImage（必需）

在构建时，此变量会解析为配方的基础映像。

示例：

```
FROM
{{{ imagebuilder:parentImage }}}
```

environments（如果指定了组件，则为必需）

此变量将解析为运行组件的脚本。

示例：

```
{{{ imagebuilder:environments }}}
```

components（可选）

Image Builder 解析容器配方中包含的组件的构建和测试组件脚本。此变量可以置于 Dockerfile 中的任意位置，在 environments 变量之后。

示例：

```
{{{ imagebuilder:components }}}
```

7. 在目标存储库一节，指定作为本教程先决条件而创建的 Amazon ECR 存储库名称。此存储库用作管道运行区域（区域 1）中分配配置的默认设置。

> **Note**
>
> 在分配之前，目标存储库必须存在于所有目标区域的 Amazon ECR 中。

8. 要继续执行下一个步骤，请选择下一步。

步骤 3：定义基础设施配置-可选

Image Builder 在您的账户中启动 EC2 实例，以自定义映像和运行验证测试。基础设施配置设置为在构建 AWS 账户 过程中将在您的中运行的实例指定基础设施详细信息。

在基础设施配置一节中，配置选项默认为 `Create infrastructure configuration using service defaults`。这将创建一个 IAM 角色和关联的实例配置文件，供构建实例用来配置您的容器映像。您还可以创建自己的自定义基础设施配置，或使用已创建的设置。有关基础设施配置设置的更多信息，请参阅《EC2 Image Builder API 参考》CreateInfrastructureConfiguration中的。

在本教程中，我们将使用默认设置。

- 要继续执行下一个步骤，请选择下一步。

步骤 4：定义分配设置-可选

分配设置由目标区域和目标 Amazon ECR 存储库名称组成。输出 Docker 映像将部署到每个区域中指定的 Amazon ECR 存储库中。

在分配设置一节中，配置选项默认为 `Create distribution settings using service defaults`。此选项会将输出 Docker 映像分配到您的管道运行区域（区域 1）的容器配方中指定的 Amazon ECR 存储库。如果您选择 `Create new distribution settings`，则可以覆盖当前区域的 ECR 存储库，并添加更多区域进行分配。

在本教程中，我们将使用默认设置。

- 要继续执行下一个步骤，请选择下一步。

步骤 5: 审核

审核一节显示您配置的所有设置。要编辑任何给定一节中的信息，请选择位于步骤一节中右上角的编辑按钮。例如，如果要更改管道名称，请选择步骤 1：管道详细信息一节中右上角的编辑按钮。

1. 查看设置后，选择创建管道来创建您的管道。
2. 如您的资源是为分配设置、基础架构配置、新配方和管道而创建，您可以在页面顶部看到成功或失败的消息。要查看资源的详细信息，包括资源标识符，请选择查看详细信息。
3. 查看资源的详细信息后，您可以通过从导航窗格中选择资源类型来查看有关其他资源的详细信息。例如，要查看新管道的详细信息，请从导航窗格中选择映像管道。如果构建成功，则您的新管道将显示在映像管道列表中。

步骤 6：清除

Image Builder 环境就像家一样，需要定期维护，以帮助您找到所需的内容，并在不费吹灰之力的情况下完成任务。请务必定期对为了测试而创建的临时资源进行清理。否则，您可能会忘记这些资源，然后再也不记得它们的用途。届时，可能还不清楚您能否安全地删除它们。

> **Tip**
>
> 为防止在删除资源时出现依赖项错误，请确保按以下顺序删除资源：
>
> 1. 映像管道
> 2. 映像配方
> 3. 所有剩余的资源

要清除您为本教程创建的资源，请执行以下步骤：

删除管道

1. 要查看在您的账户下创建的构建管道列表，请从导航窗格中选择映像管道。
2. 导航到 镜像管道 页面，然后选中要删除的管道名称旁边的复选框。
3. 在映像管道面板顶部的操作菜单中，选择删除。
4. 若要确认删除，请在框中输入 `Delete`，然后选择删除。

删除容器配方

1. 要查看在您的账户下创建的容器配方列表，请从导航窗格中选择容器配方。
2. 选择配方名称旁边的复选框以选择想要删除的配方。
3. 在容器配方面板顶部的操作菜单中，选择删除配方。
4. 若要确认删除，请在框中输入 `Delete`，然后选择删除。

删除基础设施配置

1. 要查看在您的账户下创建的基础设施配置列表，请从导航窗格中选择基础设施配置。
2. 选择配置名称旁边的复选框以选择想要删除的基础架构配置。
3. 在基础设施配置面板的顶部，选择删除。

4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除分配设置

1. 要查看在您的账户下创建的分配设置列表，请从导航窗格中选择分配设置。
2. 选择配置名称旁边的复选框以选择您为本教程创建的分配设置。
3. 在分配设置面板的顶部，选择删除。
4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除映像

请按照以下步骤验证您是否已删除从教程管道中创建的所有映像。本教程不太可能创建映像，除非根据构建计划，自您创建管道以来已经过了足够的时间，使得它运行。

1. 要查看在您的账户下创建的映像列表，请从导航窗格中选择映像。
2. 对于要移除的映像，选择映像版本。此时将打开映像构建版本页面。
3. 选中要删除的任何图像的版本旁边的复选框。您一次可以选择多个映像版本。
4. 在映像构建版本面板的顶部，选择删除版本。
5. 若要确认删除，请在框中输入 Delete，然后选择删除。

教程：创建带有输入参数的自定义组件

您可以直接从 Image Builder 控制台、Im EC2 age Builder API 或 Image Builder API 管理图像生成器组件 AWS CLI，包括创建和设置组件参数 SDKs。在本节中，我们将介绍在组件中创建和使用参数，以及在运行时通过 Image Builder 控制台和 AWS CLI 命令设置组件参数。

> **Important**
>
> 组件参数是纯文本值，并且已记录在 AWS CloudTrail中。我们建议您使用 AWS Secrets Manager 或 P AWS Systems Manager arameter Store 来存储您的密钥。有关 Secrets Manager 的更多信息，请参阅AWS Secrets Manager 用户指南中的什么是 Secrets Manager?。有关 AWS Systems Manager Parameter Store 的更多信息，请参阅 AWS Systems Manager 用户指南中的 AWS Systems Manager Parameter Store。

在 YAML 组件文档中使用参数

要构建组件，必须提供 YAML 或者 JSON 应用程序组件文档。该文档包含了在您为提供映像自定义而定义的阶段和步骤中运行的代码。引用组件的配方可以设置参数以在运行时自定义值，如果参数未设置为特定值，则默认值将生效。

使用输入参数创建组件文档

本节将介绍如何在 YAML 组件文档中定义和使用输入参数。

要在 Image Builder 构建或测试实例中创建使用参数并运行命令的 YAML 应用程序组件文档，请按照与您的映像操作系统匹配的步骤进行操作：

Linux

创建 YAML 组件文档

使用文件编辑工具创建组件文档文件。文档示例使用名为 *hello-world-test.yaml* 的文件，其中包含以下内容：

```
# Document Start
#
name: "HelloWorldTestingDocument-Linux"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
```

```
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> 在代码环境中使用像在线 YAML Validat 或 YAML lint 扩展这样的工具来验证 YAML 格式是否正确。

Windows

创建 YAML 组件文档

使用文件编辑工具创建组件文档文件。文档示例使用名为 *hello-world-test.yaml* 的文件，其中包含以下内容：

```
# Document Start
#
name: "HelloWorldTestingDocument-Windows"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
```

```
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> 在代码环境中使用像在线 YAML Validat 或 YAML lint 扩展这样的工具来验证 YAML 格式是否正确。

macOS

创建 YAML 组件文档

使用文件编辑工具创建组件文档文件。文档示例使用名为 *hello-world-test.yaml* 的文件，其中包含以下内容：

```
# Document Start
#
name: "HelloWorldTestingDocument-macOS"
```

```
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> 在代码环境中使用像在线 YAML Validat 或 YAML lint 扩展这样的工具来验证 YAML 格式是否正确。

有关 AWSTOE 应用程序组件文档的阶段、步骤和语法的更多信息，请参阅在 AWSTOE中使用文档。有关参数及其要求的更多信息，请参阅在 AWSTOE中定义和引用变量页面的 参数 部分。

从 YAML 组件文档中创建组件

无论您使用什么方法来创建 AWSTOE 组件，YAML 应用程序组件文档都必须作为基准。

- 要使用 Image Builder 控制台直接从 YAML 文档创建组件，请参阅通过控制台创建自定义组件。
- 要使用 Image Builder create-component 命令从命令行创建组件，请参阅从中创建自定义组件 AWS CLI。将这些示例中的 YAML 文档名称替换为您的 Hello World YAML 文档的名称 (*hello-world-test.yaml*)。

通过控制台在 Image Builder 配方中设置组件参数

对于映像配方和容器配方，设置组件参数的作用相同。创建新配方或配方的新版本时，可以从构建组件和测试组件列表中选择要包含的组件。组件列表包括适用于您为映像选择的基本操作系统的组件。

选择组件后，该组件将显示在组件列表正下方的选定组件部分中。将显示每个选定组件的配置选项。如果您的组件定义了输入参数，则它们将显示为名为输入参数的可扩展部分。

为组件定义的每个参数都会显示以下参数设置：

- 参数名称（不可编辑）- 参数的名称。
- 描述（不可编辑）- 参数描述
- 类型（不可编辑）- 参数值的数据类型。
- 值 – 参数的值。如果您在此配方中首次使用此组件，并且已为该输入参数定义了默认值，则默认值以灰色文本显示在值框中。如果未输入其他值，Image Builder 将使用默认值。

在食谱中使用基本图像参数

在创建图像自定义方法时，有几种方法可以识别您一开始使用的基础图片。如果您为基础映像指定亚马逊系统映像 (AMI) ID，并且该基础映像已更新，则其 AMI ID 可能会更改，您需要更新配方才能匹配。

您可以定义 AWS Systems Manager 参数存储参数（SSM 参数）来存储基础映像 AMI ID 的值，然后使用该参数在配方中指定基础映像，而不必在每次更改基础映像 ID 时都更改配方。对于 AWS 托管 AMIs，您可以使用最新版本的公共参数。

本教程将引导您完成创建 AMI ID 参数并在图像配方中使用该参数的过程。本教程中的 Image Builder 步骤基于控制台。

内容

步骤 1：查找或创建参数存储参数

此步骤的过程取决于您为基础映像指定的 AMI 类型。对于 AWS 托管 AMIs，您可以使用引用当前版本的公共参数。有些参数可能并非全部可用 AWS 区域。

首先，打开与您的 AMI 对应的选项卡。

AWS managed AMI

如果您的基础映像是 AWS 托管 AMI，则可以使用公共参数指定 AMI ID，而不必创建自己的参数。要查找您的 AMI 的公共参数，请参阅AWS Systems Manager 用户指南中的发现公共参数。

Custom AMI

要创建 AMI ID 参数，请使用控制台、或按照在 S ystems Manager 中创建参数存储库参数的说明进行操作 PowerShell。 AWS CLI提供以下值以确保参数值为 AMI ID。

参数层：`Standard`

类型：`String`

数据类型：选择`aws:ec2:image`。当您指定此类型时，系统会验证输入的值以确保它是 AMI ID。

值：输入有效的 AMI ID（例如***ami-1234567890abcdef1***）。

步骤 2：配置 IAM 权限（可选）

要使用 Systems Manager 参数存储参数（SSM 参数），无论是公共的还是私有的，都必须在 IAM 角色中指定以下 Systems Manager 参数存储操作，并将该参数列为资源。Image Builder 服务相关角色授予获取公共参数或获取或更新带`/imagebuilder/`前缀的私有参数的权限。对于没有该前缀的私有参数，您可以为执行角色添加权限。

- `ssm:GetParameter`— 此操作允许您使用 SSM 参数在食谱中指定基础图像。

- `ssm:PutParameter`— 此操作允许您在分发期间将输出 AMI ID 存储在 SSM 参数中。策略定义看起来相同，但本教程在示例策略中不包括放置操作。

1. 创建自定义角色（可选）

   在中创建管道或使用 create-image 命令时 AWS CLI，只能指定一个 Image Builder 执行角色。如果您定义了 Image Builder 工作流程执行角色，则需要向该角色添加任何其他功能权限。否则，您将创建一个包含所需权限的新自定义角色。如果您已经定义了自定义执行角色，则可以跳过此步骤。

   按照《AWS Identity and Access Management 用户指南》中创建角色向 AWS 服务委派权限的过程进行操作。

2. 为您的自定义角色添加权限

   要向自定义角色添加 SSM 参数权限，请按照《AWS Identity and Access Management 用户指南》中的 “更新角色权限策略” 流程进行操作。

   以下策略示例显示了使用在您的账户中创建的参数的`ssm:GetParameter`操作。

   JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "PrivateParameterCustomRole",
   "Effect": "Allow",
   "Action": "ssm:GetParameter",
   "Resource": "arn:aws:ssm:*:111122223333:parameter/parameter-name"
  }
 ]
}
```

有关公共参数资源的更多信息，请参阅AWS Systems Manager 用户指南中的调用 AMI 公共参数。

步骤 3：创建使用参数的图像配方

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。

2. 选择 “图像配方”，然后从列表页中选择 “创建图像配方”。
3. 填写 “基础图片” 部分，如下所示：

   a. 选择 “使用自定义 AMI” 选项。这将显示其他字段，您可以在其中输入 AMI ID 或包含 AMI ID 的 SSM 参数。

   b. 选择 SSM 参数选项。

   c. 在 SSM 参数字段中，输入您在步骤 1 中创建的参数的参数名称或 Amazon 资源名称 (ARN)。如果输入名称，则控制台中将没有前缀。
4. 根据需要完成剩余的配方配置。

> **Note**
>
> 如果您通过其他接口（例如）设置父映像 AWS CLI，则参数名称的前缀必须为`ssm:`（例如，`ssm:`*`/ImageBuilder-Tutorial/BaseAMI`*。

使用组件自定义 Image Builder 映像

Image Builder 使用 AWS Task Orchestrator and Executor (AWSTOE) 组件管理应用程序来协调复杂的工作流程。基于 YAML 文档构建和测试与 AWSTOE 应用程序配合使用的组件，这些文档定义了用于自定义或测试映像的脚本。对于 AMI 映像，Image Builder 会在其 Amazon EC2 构建和测试实例上安装 AWSTOE 组件和组件管理应用程序。对于容器镜像，组件和 AWSTOE 组件管理应用程序安装在正在运行的容器内。

Image Builder 用于 AWSTOE 执行所有实例上的活动。在运行 Image Builder 命令或使用 Image Builder 控制台 AWSTOE 时，无需进行其他设置即可进行交互。

> **Note**
>
> 当 Amazon 托管的组件达到其支持的生命周期时，该组件将不再维护。在此情况发生前大约四周，任何正在使用该组件的账户都会收到通知，以及他们账户中受其 AWS Health Dashboard 影响的配方的列表。要了解更多信息 AWS Health，请参阅《AWS Health 用户指南》。

构建新映像的工作流程期

用于构建新映像的 Image Builder 工作流程包括以下两个不同的时期。

1. 构建期（快照前）— 在构建期，您可以对运行您的基础映像的 Amazon EC2 构建实例进行更改，以创建新映像的基准。例如，您的配方可以包含安装应用程序或修改操作系统防火墙设置的组件。

   在构建期运行组件文档中的以下阶段：

   - build
   - 验证

   成功完成此阶段后，Image Builder 会创建快照或容器映像，以供测试期和后续时期使用。

2. 测试阶段（快照后）– 在测试阶段，创建 AMI 的映像与容器映像之间存在一些差异。对于 AMI 工作流，Image Builder 从其创建的快照启动 EC2 实例，作为构建阶段的最后一步。在新实例上运行测试以验证设置并确保该实例按预期运行。对于容器工作流，测试在用于构建的同一实例上运行。

   在映像构建测试期，对于配方中包含的每个组件都将运行组件文档中的以下阶段：

   - 测试

此组件期适用于“构建”和“测试”这两种组件类型。成功完成此时期后，Image Builder 可以根据快照或容器映像创建和分配最终映像。

> **Note**
>
> 虽然 AWSTOE 应用程序框架允许您在组件文档中定义许多阶段，但 Image Builder 对运行哪些阶段以及运行这些阶段有严格的规定。要使组件在映像构建期运行，组件文档必须至少定义以下阶段之一：`build` 或 `validate`。要使组件在映像测试期运行，组件文档必须定义 `test` 阶段，而不能定义其他阶段。
>
> 由于 Image Builder 独立运行各个时期，因此组件文档中的链接引用不能跨越时期边界。您不能将值从在构建期运行的阶段链接到在测试期运行的阶段。但是，您可以为预期目标定义输入参数，并通过命令行传入值。有关在 Image Builder 配方中设置组件参数的更多信息，请参阅教程：创建带有输入参数的自定义组件。

为了帮助对构建或测试实例进行故障排除，请 AWSTOE 创建一个包含输入文档和日志文件的日志文件夹，以跟踪组件每次运行时发生的情况。如果您在管道配置中配置了 Amazon S3 存储桶，则日志也会写入那里。有关 YAML 文档和日志输出的更多信息，请参阅 使用 AWSTOE 组件文档框架创建自定义组件。

> **Tip**
>
> 当需要跟踪多个组件时，标记可以帮助您根据分配给特定组件或版本的标签来识别该组件或版本。有关使用 Image Builder 命令为资源添加标签的更多信息 AWS CLI，请参阅本指南的标记资源部分。

本节介绍如何使用 Image Builder 控制台或 AWS CLI中的命令列出、查看、创建和导入组件。

主题

- 列出并查看组件详细信息
- 使用 AWS Marketplace 组件自定义您的图像
- 使用托管组件自定义 Image Builder 映像
- 为 Image Builder 映像开发自定义组件
- Image Builder 如何使用 AWS Task Orchestrator and Executor 应用程序管理组件

列出并查看组件详细信息

本节介绍如何查找您在 EC2 Image Builder 配方中使用的组件的信息并查看其详细信息。

组件详细信息

- 列出 Image Builder 组件
- 列出来自的组件编译版本 AWS CLI
- 从中获取组件详细信息 AWS CLI
- 从中获取组件策略的详细信息 AWS CLI

列出 Image Builder 组件

您可以使用以下方法之一列出和筛选 Image Builder 组件。

AWS 管理控制台

要在中显示组件列表 AWS 管理控制台，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中选择组件。默认情况下，Image Builder 会显示您的账户拥有的组件列表。
3. 您可以选择根据组件所有权进行筛选。要查看您不拥有但可以访问的组件，请展开所有者类型下拉列表并选择其中一个值。所有者类型列表位于搜索栏中的搜索文本框旁边。可以选择以下值：
   - AWS Marketplace— 与 AWS Marketplace 产品订阅直接关联的组件。
   - 快速入门（Amazon 托管）— Amazon 创建和维护的公开可用组件。
   - 我拥有的 — 您创建的组件。这是默认选择。
   - 与我共享 — 其他人通过其账户创建并与您共享的组件。
   - 第三方管理 — 您订阅的第三方拥有的组件。 AWS Marketplace

AWS CLI

以下示例说明如何使用 list-components 命令返回您的账户拥有的 Image Builder 组件列表。

```
aws imagebuilder list-components
```

您可以选择根据组件所有权进行筛选。所有者属性定义您要列出的组件的拥有人。默认情况下，此请求会返回您的账户拥有的组件列表。要按组件所有者筛选结果，请在运行 list-components 命令时使用 --owner 参数指定以下值之一。

组件所有者的值

- AWSMarketplace
- Amazon
- Self
- Shared
- ThirdParty

以下示例显示带有 --owner 参数的 list-components 命令，用于筛选结果。

```
aws imagebuilder list-components --owner Self
{
    "requestId": "012a3456-b789-01cd-e234-fa5678b9012b",
    "componentVersionList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/sample-
component01/1.0.0",
            "name": "sample-component01",
            "version": "1.0.0",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2020-09-24T16:58:24.444Z"
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/sample-
component01/1.0.1",
            "name": "sample-component01",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2021-07-10T03:38:46.091Z"
        }
    ]
}
```

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

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

```
aws imagebuilder list-components --owner ThirdParty
```

列出来自的组件编译版本 AWS CLI

以下示例说明了如何使用 list-component-build-versions 命令列出具有特定语义版本的组件构建版本。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

```
aws imagebuilder list-component-build-versions --component-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:component/example-component/1.0.1
{
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "componentSummaryList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
examplecomponent/1.0.1/1",
            "name": "examplecomponent",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "description": "An example component that builds, validates and tests an
 image",
            "changeDescription": "Updated version.",
            "dateCreated": "2020-02-19T18:53:45.940Z",
            "tags": {
                "KeyName": "KeyValue"
            }
        }
    ]
}
```

从中获取组件详细信息 AWS CLI

以下示例说明在指定组件的 Amazon 资源名称（ARN）时如何使用 get-component 命令获取组件详细信息。

```
aws imagebuilder get-component --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/example-component/1.0.1/1
   {
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11112",
    "component": {
        "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
examplecomponent/1.0.1/1",
        "name": "examplecomponent",
        "version": "1.0.1",
        "type": "BUILD",
        "platform": "Linux",
        "owner": "123456789012",
        "data": "name: HelloWorldTestingDocument\ndescription: This is hello world
 testing document... etc.\"\n",
        "encrypted": true,
        "dateCreated": "2020-09-24T16:58:24.444Z",
        "tags": {}
    }
}
```

从中获取组件策略的详细信息 AWS CLI

以下示例说明了在指定组件的 ARN 时如何使用 get-component-policy 命令获取组件策略的详细信息。

```
aws imagebuilder get-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/example-component/1.0.1
```

使用 AWS Marketplace 组件自定义您的图像

除了独立软件供应商 (ISVs) 创建的大量图像外，还 AWS Marketplace 提供可用于自定义自己的 Image Builder 映像的组件。您必须先订阅这些 AWS Marketplace 组件，然后才能在图像配方中使用它们来构建新映像。

当您在图像配方中指定 AWS Marketplace 组件时，Image Builder 会验证订阅并执行依赖关系检查，以确保您拥有使用该组件所需的资源。验证成功后，Image Builder 会为组件及其构件创建安全下载内容，供映像管道构建使用。

探索 AWS Marketplace 组件

您可以从 Image Builder 控制台的 “发现产品” 页面中查找要在配方中使用的 AWS Marketplace 软件组件，如下所示。

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择AWS Marketplace部分中的发现产品。
3. 选择 Components（组件）选项卡。此选项卡列出了使用配送选项的所有 AWS Marketplace 产品，其中包括中的相关组件 AWS Marketplace。
4. 要搜索包含组件的特定软件产品，可以在搜索栏中输入部分名称或按`Status`、`Operating System`、`Publisher`、或进行筛选`Categories`。搜索栏还包含搜索结果的分页控件。

结果

每种 AWS Marketplace 产品都有自己的详情面板，其中包含以下信息。

AWS Marketplace 产品名称和徽标

软件产品名称链接到中的产品详细信息 AWS Marketplace。您可以选择链接以了解有关该产品的更多信息 AWS Marketplace。或者，如果您已经完成研究，则可以查看订阅选项摘要，并使用查看订阅选项按钮直接从搜索结果中进行订阅。

版本

它包含该组件的主版本。

操作系统

设计用于运行该组件的操作系统。

出版商

组件的发布者。它链接到中的出版商详情页面 AWS Marketplace。此时将在浏览器的新选项卡中打开发布者详情页面。

类别

适用于该组件的一个或多个 AWS Marketplace 产品类别。

状态

显示您是否订阅了此产品。如果您尚未订阅，则可以选择 “查看订阅选项” 以查看 AWS Marketplace 产品的订阅选项摘要，也可以选择直接从 Image Builder 控制台进行订阅。

关联组件

如果您的订阅中包含该 AWS Marketplace 产品的一个或多个版本，则这些版本将显示在 “关联组件” 部分中。该部分最初处于折叠状态，并显示关联组件的数量。您可以展开该部分以查看更多详细信息。

> **Note**
>
> 与其 AWS Marketplace 图像产品关联的互联网安全中心 (CIS) 组件未显示在 D iscover 产品的结果中。如果您订阅了他们的图像产品，则关联的组件将显示在 “订阅” 页面中，并在 “创建图像配方” 对话框中显示为第三方组件。有关该组件的更多信息，请参阅CIS 强化组件。

订阅 AWS Marketplace 组件

找到包含要在食谱中使用的组件 AWS Marketplace 的产品后，您可以直接从 Image Builder 控制台订阅该产品，如下所示，也可以从 AWS Marketplace 控制台订阅。

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择AWS Marketplace部分中的发现产品。
3. 选择 Components（组件）选项卡。此选项卡列出了使用配送选项的所有 AWS Marketplace 产品，其中包括中的相关组件 AWS Marketplace。
4. 要搜索特定 AWS Marketplace 产品，请在搜索栏中输入部分名称。如果您知道出版商，但不知道确切的产品名称或拼写方式，则也可以筛选`Publisher`以获取出版商提供的产品列表。
5. 从结果列表中选择要订阅的产品，然后选择查看订阅选项。这显示了该 AWS Marketplace 产品的订阅选项摘要。
6. 选择 “订阅”，无需离开 Image Builder 控制台即可订阅产品。系统会通知您订阅正在处理中。订阅后，状态将更新为。`Subscribed`

有关您当前订阅的 AWS Marketplace 产品的更多信息，请参阅中所AWS Marketplace Image Builder 中的订阅述的控制台步骤。

在 Image Builder 图像配方中使用 AWS Marketplace 组件

您可以像使用其他类型的 AWS Marketplace 组件一样在 Image Builder 图像配方中使用组件。对于与 AWS Marketplace 图片产品关联的大多数组件，所有权类别为AWS Marketplace。例如，要使用您已订阅 AWS Marketplace 的产品中的构建组件，请选择 “添加构建组件”，然后AWS Marketplace从列表中进行选择。这将在控制台界面的右侧打开一个列出 AWS Marketplace 组件的选择面板。

> **Note**
>
> 如果您正在寻找 CIS 强化组件Third party managed，请从所有权列表中选择，而不是。AWS Marketplace

有关如何为组件选择、排列和配置参数的更多信息，请参阅创建映像配方的新版本。

使用托管组件自定义 Image Builder 映像

托管组件由第三方组织创建 AWS，有时是与第三方组织合作创建，例如互联网安全中心 (CIS)。当您在映像或容器配方中使用托管组件时，Amazon 会提供已应用补丁和其他更新的最新组件版本。要获取组件列表或获取组件信息，请参阅列出并查看组件详细信息。

以下精选 AWS 托管组件列表包括一个组件，当你订阅 AMIs 经过强化的 CIS 时，你可以使用该 AWS Marketplace组件。

精选组件

- 为 Image Builder Windows 映像安装 Distributor 软件包托管组件应用程序
- CIS 强化组件
- Image Builder 的 Amazon 托管 STIG 固化组件

为 Image Builder Windows 映像安装 Distributor 软件包托管组件应用程序

AWS Systems Manager Distributor 可以帮助您将软件打包并发布到 AWS Systems Manager 托管节点。您可以打包和发布自己的软件，也可以使用 Distributor 查找和发布 AWS提供的代理软件包。有关 Systems Manager Distributor 的更多信息，请参阅《AWS Systems Manager 用户指南》中的 AWS Systems Manager Distributor。

Distributor 的托管组件

以下 Image Builder 托管组件使用 AWS Systems Manager Distributor 在 Windows 实例上安装应用程序包。

- `distributor-package-windows` 托管组件使用 AWS Systems Manager Distributor 来安装您在 Windows 映像构建实例上指定的应用程序包。要在将此组件包含在配方中时配置参数，请参阅配置 `distributor-package-windows` 为独立组件。
- 该`aws-vss-components-windows`组件使用 Dist AWS Systems Manager ributor 在你的 Windows 映像生成实例上安装`AwsVssComponents`软件包。要在将此组件包含在配方中时配置参数，请参阅配置 `aws-vss-components-windows` 为独立组件。

有关如何在 Image Builder 配方中使用托管组件的更多信息，请参阅适用于映像配方的 创建映像配方的新版本 或适用于容器配方的 创建新版本的容器配方。有关 `AwsVssComponents` 软件包的更多信息，请参阅《Amazon EC2 用户指南》中的创建 VSS 应用程序一致性快照。

先决条件

在使用依赖 Systems Manager Distributor 的 Image Builder 组件安装应用程序包之前，必须确保满足以下先决条件。

- 使用 Systems Manager Distributor 在您的实例上安装应用程序包的 Image Builder 组件需要获得调用 Systems Manager API 的权限。在您使用 Image Builder 配方中的组件之前，必须创建用于授予权限的 IAM policy 和角色。要配置权限，请参阅 配置 Systems Manager Distributor 权限。

> **Note**
>
> Image Builder 目前不支持重启实例的 Systems Manager Distributor 软件包。例如，`AWSNVMe`、`AWSPVDrivers` 和 `AwsEnaNetworkDriver` Distributor 软件包会重启实例，因此是不允许的。

配置 Systems Manager Distributor 权限

`distributor-package-windows` 组件和其他使用它的组件（例如 `aws-vss-components-windows`）需要获得构建实例的额外权限才能运行。构建实例必须能够调用 Systems Manager API 才能开始安装 Distributor 并轮询结果。

按照中的这些步骤创建自定义 IAM 策略和角色，该 AWS 管理控制台 策略和角色授予 Image Builder 组件从构建实例安装 Systems Manager Distributor 包的权限。

步骤 1：创建策略

为 Distributor 权限创建 IAM policy。

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 策略，然后选择 创建策略。
3. 在创建策略页面上，选择 JSON 选项卡，然后将默认内容替换为以下 JSON 策略，根据需要替换分区、区域和账户 ID，或者使用通配符。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDistributorSendCommand",
            "Effect": "Allow",
            "Action": "ssm:SendCommand",
            "Resource": [
                "arn:aws:ssm:*::document/AWS-ConfigureAWSPackage",
                "arn:aws:ec2:*:111122223333:instance/*"
            ]
        },
        {
            "Sid": "AllowGetCommandInvocation",
            "Effect": "Allow",
            "Action": "ssm:GetCommandInvocation",
            "Resource": "*"
        }
    ]
}
```

4. 选择查看策略。
5. 对于名称，请输入一个名称以标识该策略，例如 *InvokeDistributor* 或您喜欢的其他名称。
6. （可选）对于描述，请输入该角色的用途描述。
7. 选择 Create policy。

步骤 2：创建角色

为 Distributor 权限创建 IAM 角色。

1. 在 IAM 控制台导航窗格中，选择 角色，然后选择 创建角色。
2. 在选择受信任实体的类型下，选择 AWS 服务。
3. 在紧靠选择将使用此角色的服务下面，选择 EC2，然后选择下一步: 权限。
4. 在选择您的使用案例下面，选择 EC2，然后选择下一步: 权限。
5. 在政策列表中，选中 Amazon 旁边的复选框SSMManagedInstanceCore。（如果需要缩小列表范围，请在搜索框中键入 SSM。）
6. 在此策略列表中，选中旁边的复选框EC2InstanceProfileForImageBuilder。（如果需要缩小列表范围，请在搜索框中键入 ImageBuilder。）
7. 选择下一步：标签。
8. （可选）添加一个或多个标签键值对，以组织、跟踪或控制此角色的访问，然后选择 下一步：审核。
9. 对于角色名称，请输入角色的名称，例如 *InvokeDistributor* 或您喜欢的其他名称。
10. （可选）对于角色描述，请将默认文本替换为该角色的用途描述。
11. 选择 Create role (创建角色)。系统将让您返回到 角色 页面。

步骤 3：向角色附加该策略

设置您的 Distributor 权限的最后一步是将您创建的 IAM policy 附加到 IAM 角色。

1. 在 IAM 控制台 Roles（角色）页面上，选择您刚刚创建的角色。将打开角色摘要页面。
2. 选择附加策略。
3. 搜索在上一过程中创建的策略，然后选中该名称旁边的复选框。
4. 选择 Attach policy（附加策略）。

对于包含使用 Systems Manager Distributor 的组件的任何映像，在 Image Builder 基础设施配置资源中使用此角色。有关更多信息，请参阅 创建基础设施配置。

配置 `distributor-package-windows` 为独立组件

要在配方中使用 `distributor-package-windows` 组件，请设置以下参数来配置要安装的程序包。

> **Note**
>
> 必须确保满足所有 先决条件，才能在配方中使用 `distributor-package-windows` 组件。

- 操作（必需）- 指定是安装还是卸载软件包。有效值包括 `Install` 和 `Uninstall`。此值默认为 `Install`。
- PackageName（必填）-要安装或卸载的分销商软件包的名称。有关有效软件包名称的列表，请参阅查找 Distributor 软件包。
- PackageVersion（可选）-要安装的分销商软件包的版本。 PackageVersion 默认为推荐版本。
- AdditionalArguments（可选）— 一个 JSON 字符串，其中包含要提供给脚本以安装、卸载或更新软件包的其他参数。有关更多信息，请参阅 Systems Manager 命令文档插件引用页面 aws:configurePackage 输入章节中的 additionalArguments。

配置 `aws-vss-components-windows` 为独立组件

在配方中使用 `aws-vss-components-windows` 组件时，可以选择设置 `PackageVersion` 参数，以使用 `AwsVssComponents` 程序包的特定版本。如果省略此参数，该组件将默认使用 `AwsVssComponents` 程序包的推荐版本。

> **Note**
>
> 必须确保满足所有 先决条件，才能在配方中使用 `aws-vss-components-windows` 组件。

查找 Distributor 软件包

亚马逊和第三方提供公共软件包，您可以通过 Systems Manager Distributor 安装这些软件包。

要在中查看可用软件包 AWS 管理控制台，请登录AWS Systems Manager 控制台并从导航窗格中选择 Dist ri butor。Distributor 页面显示可供您使用的所有软件包。有关使用列出可用软件包的更多信息 AWS CLI，请参阅《AWS Systems Manager 用户指南》中的查看软件包（命令行）。

您还可以创建自己的私有 Systems Manager Distributor 软件包。有关更多信息，请参阅 《AWS Systems Manager 用户指南》中的创建软件包。

CIS 强化组件

Center for Internet Security (CIS) 是一个以社区为导向的非营利组织。他们的网络安全专家共同制定 IT 安全指南，以保护公共和私人组织免受网络威胁。他们全球公认的一套最佳实践，即 CIS 基准，可帮助世界各地的 IT 组织安全地配置其系统。有关热门文章、博客文章、播客、网络研讨会和白皮书，请参阅 Center for Internet Security 网站上的 CIS Insights（CIS 洞见）。

CIS 基准

CIS 创建并维护一套配置指南，即 CIS 基准，为特定技术（包括操作系统、云平台、应用程序、数据库等）提供配置最佳实践。CIS 基准被 PCI DSS、HIPAA、DoD Cloud Computing SRG、FISMA、DFARS 和 FEDRAMP 等组织和标准认可为行业标准。要了解更多信息，请参阅 Center for Internet Security 网站上的 CIS Benchmarks（CIS 基准）。

CIS 强化组件

在中订阅 CIS 加固映像时 AWS Marketplace，您还可以访问相关的强化组件，该组件运行脚本来为您的配置强制执行 CIS 基准第 1 级指南。CIS 组织拥有并维护 CIS 强化组件，以确保它们反映最新指南。

> **Note**
>
> CIS 加固组件未遵循 Image Builder 配方中的标准组件排序规则。CIS 强化组件始终最后运行，以确保基准测试针对您的输出映像运行。

Image Builder 的 Amazon 托管 STIG 固化组件

安全技术实施指南 (STIGs) 是国防信息系统局 (DISA) 为保护信息系统和软件而制定的配置强化标准。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。

Image Builder 提供 STIG 强化组件，可帮助您更有效地构建符合基准 STIG 标准的兼容映像。这些 STIG 组件扫描错误的配置并运行修复脚本。使用符合 STIG 要求的组件不会收取额外的费用。

> **Important**
>
> 除了少数例外，除非通过参数指定，否则 STIG 加固组件不会安装第三方软件包。如果实例上已经安装了第三方软件包，并且 Image Builder 支持该软件包，则强化组件会应用这些软件包。 STIGs

本页列出了 Image Builder 支持的所有应用于 Image Builder 在您构建和测试新映像时启动的 EC2 实例的 STIG。如果要对映像应用其他 STIG 设置，可以创建一个自定义组件来对其进行配置。有关自定义组件以及如何创建它们的更多信息，请参阅 使用组件自定义 Image Builder 映像。

创建映像时，STIG 强化组件会记录是否 STIGs 已应用或跳过支持。我们建议您查看使用 STIG 强化组件的映像的 Image Builder 日志。有关如何访问和查看 Image Builder 日志的更多信息，请参阅 管道构建故障排除。

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）

  包括可能导致机密性、可用性或完整性丢失的任何漏洞，但可以减轻风险。
- 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 强化组件

AWSTOE Windows STIG 强化组件专为独立服务器而设计，并应用本地组策略。符合 STIG 的强化组件会安装和更新国防部 (DoD) 证书。它们还会删除不必要的证书，以维持 STIG 合规性。目前，以下版本的 Windows Server 支持 STIG 基准：2012 R2、2016、2019、2022 和 2025。

本节列出了每个 Windows STIG 强化组件的当前设置，然后是版本历史记录日志。

Windows STIG 低（第三类）

以下列表包含强化组件适用于您的基础架构的 STIG 设置。如果支持的设置不适用于您的基础架构，则强化组件会跳过该设置并继续向前。例如，某些 STIG 设置可能不适用于独立服务器。组织特定的策略也可能影响强化组件适用于哪些设置，如针对管理员查看文档设置的要求。

有关 Windows 的完整列表 STIGs，请参阅STIGs 文档库。有关如何查看完整列表的信息，请参阅 STIG 查看工具。

- Windows Server 2025 STIG 第 1 版 1

  V-278082、V-278083、V-278084、V-278085、V-278098、V-278104、V-278110 和 V-278231

- Windows Server 2022 STIG 第 2 版 7

  V-254335、V-254336、V-254337、V-254338、V-254351、V-254357、V-254363 和 V-254481
- Windows Server 2019 STIG 版本 3 版本 7

  V-205691、V-205819、V-205858、V-205859、V-205860、V-205870、V-205871 和 V-205923
- Windows Server 2016 STIG 版本 2 发行版 10

  V-224916、V-224917、V-224918、V-224919、V-224931、V-224942 和 V-225060
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5

  V-225250、V-225318、V-225319、V-225324、V-225327、V-225328、V-225330、V-225331、V-225332 和 V-225537
- 微软.NET Framework 4.0 STIG 第 2 版 7

  类别 III 漏洞 的 Microsoft .NET Framework 未应用 STIG 设置。
- Windows Firewall STIG 版本 2 发行版 2

  V-241994、V-241995、V-241996、V-241999、V-242000、V-242001、V-242006、V-242007 和 V-242008
- Internet Explorer 11 STIG 版本 2 版本 6

  V-223056 和 V-223078
- 微软 Edge STIG 第 2 版 4（仅限 Windows Server 2022 和 2025）

  V-235727、V-235731、V-235751、V-235752 和 V-235765
- 微软 Defender STIG 第 2 版 7

  对于第 III 类漏洞，没有将 STIG 设置应用于 Microsoft 防病毒软件。

Windows STIG 中等（第二类）

以下列表包含强化组件适用于您的基础架构的 STIG 设置。如果支持的设置不适用于您的基础架构，则强化组件会跳过该设置并继续向前。例如，某些 STIG 设置可能不适用于独立服务器。组织特定的策略也可能影响强化组件适用于哪些设置，如针对管理员查看文档设置的要求。

有关 Windows 的完整列表 STIGs，请参阅STIGs 文档库。有关如何查看完整列表的信息，请参阅 STIG 查看工具。

> **Note**
>
> Windows STIG Medium 强化组件包括所有列出的 AWSTOE 适用于 Windows STIG Low 强化组件的 STIG 设置，以及专门针对第二类漏洞列出的 STIG 设置。

- Windows Server 2025 STIG 第 1 版 1

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-278015、V-278016、V-278019、V-278020、V-278021、V-278022、V-278023、V-278024、V-278025
  V-278220 V-278221 V-278222 V-278223
  V-278245、V-278247、V-278248、V-278249、V-278251、V-278252、V-278253、V-278254、V-278255
  278261、V-278262、V-279916、V-279917、V-279918、V-279919、V-279920、V-279921、V-279921、
  和 V-279923 V-278226 V-278227 V-278228 V-278229 V-278230 V-278232 V-278233 V-278234
  V-278235

- Windows Server 2022 STIG 第 2 版 7

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-254247、V-254269、V-254270、V-254271、V-254272、V-254273、V-254274、V-254275、V-254276
  44439、V-2544440、V-254442、V-254444、V-254444、V-254447、V-254448、V-254449、V-254451、
  4462，V-254463、V-254464、V-254468、V-254470、V-254471、V-254472、V-254473
  V-254476 V-254477 V-254478
  V-254479，V-254503、V-254504、V-254505、V-254506、V-254507、V-254508、V-254509、V-254510
  8946、V-278947、V-278948 和 V-278949 V-254480 V-254482 V-254483 V-254484 V-254485
  V-254486 V-254487 V-254488 V-254489

- Windows Server 2019 STIG 版本 3 版本 7

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-205625、V-205626、V-205627、V-205629、V-205630、V-205633、V-205634、V-205636、V-205637
  18，V-205719、V-205720、V-205722、V-205730、V-205731、V-205733、V-205747、V-205749、V-205
  V-205830 V-205832 V-205833 V-205835 V-205836
  V-205837、、、f-205868、V-205869、V-205872、V-205872、V-205873、V-205874、V-2059909、V-20
  40 和 V-278941 V-205838 V-205842 V-205861 V-205863 V-205865 V-205866 V-205867

- Windows Server 2016 STIG 版本 2 发行版 10

包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

V-224850、V-224851、V-224852、V-224853、V-224854、V-224855、V-224856、V-224858、V-224859 V-224968、V-224969、V-224987、V-224988、V-224989、V-224995、V-224996、V-224997、V-224999 V-225089、V-225092、V-225093 和 V-257502

- Windows Server 2012 R2 MS STIG 版本 3 发行版 5

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-225239、V-225259、V-225260、V-225261、V-225263、V-225264、V-225265、V-225266、V-225267 和 V-225574

- 微软.NET Framework 4.0 STIG 第 2 版 7

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-225223、V-225230、V-225235 和 V-225238

- Windows Firewall STIG 版本 2 发行版 2

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-241989、V-241990、V-241991、V-241993、V-241998、V-242003、V-242004 和 V-242005

- Internet Explorer 11 STIG 版本 2 版本 6

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-223015、V-223017、V-223018、V-223019、V-223020、V-223021、V-223022、V-223023、V-223024 V-223142 V-223143 V-223144 V-223145 V-223146 V-223147 V-223148 V-223149 V-250540 V-250541

- 微软 Edge STIG 第 2 版 4（仅限 Windows Server 2022 和 2025）

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

  V-235720、V-235721、V-235723、V-235724、V-235725、V-235726、V-235728、V-235729、V-235730 和 V-246736

- 微软 Defender STIG 第 2 版 7

  包括强化组件适用于类别 III（低级）漏洞的所有支持的 STIG 设置，以及：

V-213427、V-213429、V-213430、V-213431、V-213432、V-213433、V-213434、V-213435、V-213436 和 V-278863

Windows STIG 高（第一类）

以下列表包含强化组件适用于您的基础架构的 STIG 设置。如果支持的设置不适用于您的基础架构，则强化组件会跳过该设置并继续向前。例如，某些 STIG 设置可能不适用于独立服务器。组织特定的策略也可能影响强化组件适用于哪些设置，如针对管理员查看文档设置的要求。

有关 Windows 的完整列表 STIGs，请参阅STIGs 文档库。有关如何查看完整列表的信息，请参阅 STIG 查看工具。

> **Note**
>
> Windows STIG High 强化组件包括所有列出的 AWSTOE 适用于 Windows STIG Low 和 Windows STIG Medium 强化组件的 STIG 设置，以及专门针对第一类漏洞列出的 STIG 设置。

- Windows Server 2025 STIG 第 1 版 1

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-278040、V-278099、V-278100、V-278101、V-278121、V-278125、V-278128 V-278196 V-278215 V-278216 V-278217 V-278219 V-278225 V-278242 V-278246 V-278250

- Windows Server 2022 STIG 第 2 版 7

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-254250、V-254293、V-254352、V-254353、V-254354、V-254374、V-254378 V-254381 V-254446 V-254466 V-254467 V-254469 V-254474 V-254475 V-254492 V-254496 V-254500

- Windows Server 2019 STIG 版本 3 版本 7

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-205653、V-205654、V-205663、V-205711、V-205713、V-205724、V-205725 V-205750 V-205753 V-205757 V-205802 V-205804 V-205805 V-205806 V-205849 V-205908、、V-205914 V-205919

- Windows Server 2016 STIG 版本 2 发行版 10

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-224831、V-224874、V-224932、V-224933、V-224934、V-224954、V-224958 V-224961 V-225025 V-225045 V-225046 V-225048 V-225053 V-225054 V-225071 V-225079 V-225091

- Windows Server 2012 R2 MS STIG 版本 3 发行版 5

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-225274、V-225354、V-225364、V-225365、V-225366、V-225390、V-225396、V-225399、V-225444 和 V-225556

- 微软.NET Framework 4.0 STIG 第 2 版 7

  包括强化组件适用于 Microsoft .NET Framework 的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置。第一类漏洞未应用额外 STIG 设置。

- Windows Firewall STIG 版本 2 发行版 2

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-241992、V-241997 和 V-242002

- Internet Explorer 11 STIG 版本 2 版本 6

  V-252910

- 微软 Edge STIG 第 2 版 4（仅限 Windows Server 2022 和 2025）

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-235758 和 V-235759

- 微软 Defender STIG 第 2 版 7

  包括强化组件适用于类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：

  V-213426、V-213428、V-213452 和 V-213453

适用于 Windows 的 STIG 版本历史记录日志

本节记录 STIG 季度更新的 Windows 强化组件版本历史记录。要查看一个季度的变化和发布的版本，请选择其标题以展开信息。

2026 年第一季度变化——2026 年 3 月 10 日：

增加了对 Windows Server 2025 的支持，并更新了所有适用 STIGs 于 2026 年第一季度的内容。

Stig-build-Windows

- Windows Server 2025 STIG 第 1 版 1
- Windows Server 2022 STIG 第 2 版 7
- Windows Server 2019 STIG 版本 3 版本 7
- Windows Server 2016 STIG 版本 2 发行版 10
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- 微软.NET Framework 4.0 STIG 第 2 版 7
- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 版本 8
- 微软 Edge STIG 第 2 版 4（仅限 Windows Server 2022 和 2025）

2025 年第三季度更改 - 2025 年 9 月 4 日（无更改）：

2025 年第三季度发行版的 Windows 组件 STIG 无更改。

2025 年第二季度更改 - 2025 年 6 月 26 日：

更新了 STIG 版本并对 2025 年第二季度发行版应用了 STIG，如下所示：

STIG-Build-Windows-Low 版本 2025.2.x

- Windows Server 2022 STIG 版本 2 发行版 4
- Windows Server 2019 STIG 版本 3 发行版 4
- Windows Server 2016 STIG 版本 2 发行版 10
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 6
- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 发行版 5
- Microsoft Edge STIG 版本 2 发行版 2（仅限 Windows Server 2022）

STIG-Build-Windows-Medium 版本 2025.2.x

- Windows Server 2022 STIG 版本 2 发行版 4
- Windows Server 2019 STIG 版本 3 发行版 4
- Windows Server 2016 STIG 版本 2 发行版 10
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 6
- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 发行版 5
- Microsoft Edge STIG 版本 2 发行版 2（仅限 Windows Server 2022）
- Defender STIG 版本 2 发行版 4

STIG-Build-Windows-High 版本 2025.2.x

- Windows Server 2022 STIG 版本 2 发行版 4
- Windows Server 2019 STIG 版本 3 发行版 4
- Windows Server 2016 STIG 版本 2 发行版 10
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 6
- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 发行版 5
- Microsoft Edge STIG 版本 2 发行版 2（仅限 Windows Server 2022）
- Defender STIG 版本 2 发行版 4

2025 年第一季度更改 - 2025 年 5 月 4 日：

更新了针对 Internet Explorer 11 STIG 版本 2 发行版 5 的 STIG，适用于 2025 年第一季度发行版的所有 STIG 组件。

- STIG-Build-Windows-Low 版本 2025.1.x
- STIG-Build-Windows-Medium 版本 2025.1.x
- STIG-Build-Windows-High 版本 2025.1.x

2024 年第四季度变化——2025 年 4 月 2 日：

更新了 STIG 版本并对 2024 年第四季度发行版应用了 STIG，如下所示：

STIG-Build-Windows-Low 版本 2024.4.0

- Windows Server 2022 STIG 版本 2 发行版 2
- Windows Server 2019 STIG 版本 3 发行版 2
- Windows Server 2016 STIG 版本 2 发行版 9
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 2
- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 发行版 5
- Microsoft Edge STIG 版本 2 发行版 2（仅限 Windows Server 2022）

STIG-Build-Windows-Medium 版本 2024.4.0

- Windows Server 2022 STIG 版本 2 发行版 2
- Windows Server 2019 STIG 版本 3 发行版 2
- Windows Server 2016 STIG 版本 2 发行版 9
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 2
- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 发行版 5
- Microsoft Edge STIG 版本 2 发行版 2（仅限 Windows Server 2022）
- Defender STIG 版本 2 发行版 4

STIG-Build-Windows-High 版本 2024.4.0

- Windows Server 2022 STIG 版本 2 发行版 2
- Windows Server 2019 STIG 版本 3 发行版 2
- Windows Server 2016 STIG 版本 2 发行版 9
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 2

- Windows Firewall STIG 版本 2 发行版 2
- Internet Explorer 11 STIG 版本 2 发行版 5
- Microsoft Edge STIG 版本 2 发行版 2（仅限 Windows Server 2022）
- Defender STIG 版本 2 发行版 4

2024 年第三季度变化——2023 年 4 月 10 日（没有变化）：

2024 年第三季度发行版的 Windows 组件 STIG 无更改。

2024 年第二季度更改 - 2024 年 5 月 10 日（无更改）：

在 2024 年第二季度发行版中，Windows 组件 STIGS 无更改。

2024 年第一季度更改 - 2024 年 2 月 6 日（无更改）：

在 2024 年第一季度发行版中，Windows 组件 STIGS 无更改。

2023 年第四季度更改 - 2023 年 12 月 4 日（无更改）：

在 2023 年第四季度发行版中，Windows 组件 STIGS 无更改。

2023 年第三季度更改 — 2023 年 10 月 4 日（无更改）：

在 2023 年第三季度发行版中，Windows 组件 STIGS 无更改。

2023 年第二季度更改 — 2023 年 5 月 3 日（无更改）：

在 2023 年第二季度发行版中，Windows 组件 STIGS 无更改。

2023 年第一季度更改 — 2023 年 3 月 27 日（无更改）：

在 2023 年第一季度发行版中，Windows 组件 STIGS 无更改。

2022 年第四季度更改 — 2023 年 2 月 1 日：

更新了 STIG 版本并对 2022 年第四季度发行版应用了 STIG，如下所示：

STIG-Build-Windows-Low 版本 2022.4.x

- Windows Server 2022 STIG 版本 1 发行版 1
- Windows Server 2019 STIG 版本 2 发行版 5
- Windows Server 2016 STIG 版本 2 发行版 5

- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 2
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 2 发行版 3
- Microsoft Edge STIG 版本 1 发行版 6（仅限 Windows Server 2022）

STIG-Build-Windows-Medium 版本 2022.4.x

- Windows Server 2022 STIG 版本 1 发行版 1
- Windows Server 2019 STIG 版本 2 发行版 5
- Windows Server 2016 STIG 版本 2 发行版 5
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 2
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 2 发行版 3
- Microsoft Edge STIG 版本 1 发行版 6（仅限 Windows Server 2022）
- Defender STIG 版本 2 发行版 4（仅限 Windows Server 2022）

STIG-Build-Windows-High 版本 2022.4.x

- Windows Server 2022 STIG 版本 1 发行版 1
- Windows Server 2019 STIG 版本 2 发行版 5
- Windows Server 2016 STIG 版本 2 发行版 5
- Windows Server 2012 R2 MS STIG 版本 3 发行版 5
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 2
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 2 发行版 3
- Microsoft Edge STIG 版本 1 发行版 6（仅限 Windows Server 2022）
- Defender STIG 版本 2 发行版 4（仅限 Windows Server 2022）

2022 年第三季度更改 — 2022 年 9 月 30 日（无更改）：

在 2022 年第三季度发行版中，Windows 组件 STIGS 无更改。

2022 年第二季度更改 — 2022 年 8 月 2 日：

更新了 STIG 版本，并对 2022 年第二季度发行版应用了 STIG。

STIG-Build-Windows-Low 版本 1.5.x

- Windows Server 2019 STIG 版本 2 发行版 4
- Windows Server 2016 STIG 版本 2 发行版 4
- Windows Server 2012 R2 MS STIG 版本 3 发行版 3
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 1 发行版 19

STIG-Build-Windows-Medium 版本 1.5.x

- Windows Server 2019 STIG 版本 2 发行版 4
- Windows Server 2016 STIG 版本 2 发行版 4
- Windows Server 2012 R2 MS STIG 版本 3 发行版 3
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 1 发行版 19

STIG-Build-Windows-High 版本 1.5.x

- Windows Server 2019 STIG 版本 2 发行版 4
- Windows Server 2016 STIG 版本 2 发行版 4
- Windows Server 2012 R2 MS STIG 版本 3 发行版 3
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 1 发行版 19

2022 年第一季度更改 — 2022 年 2 月 8 日（无更改）：

在 2022 年第一季度发行版中，Windows 组件 STIGS 无更改。

2021 年第四季度更改 — 2021 年 12 月 20 日：

更新了 STIG 版本，并对 2021 年第四季度发行版应用了 STIG。

STIG-Build-Windows-Low 版本 1.5.x

- Windows Server 2019 STIG 版本 2 发行版 3
- Windows Server 2016 STIG 版本 2 发行版 3
- Windows Server 2012 R2 MS STIG 版本 3 发行版 3
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 1 发行版 19

STIG-Build-Windows-Medium 版本 1.5.x

- Windows Server 2019 STIG 版本 2 发行版 3
- Windows Server 2016 STIG 版本 2 发行版 3
- Windows Server 2012 R2 MS STIG 版本 3 发行版 3
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 1 发行版 19

STIG-Build-Windows-High 版本 1.5.x

- Windows Server 2019 STIG 版本 2 发行版 3
- Windows Server 2016 STIG 版本 2 发行版 3
- Windows Server 2012 R2 MS STIG 版本 3 发行版 3
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 2 发行版 1
- Internet Explorer 11 STIG 版本 1 发行版 19

2021 年第三季度更改 — 2021 年 9 月 30 日：

更新了 STIG 版本，并对 2021 年第三季度发行版应用了 STIG。

STIG-Build-Windows-Low 版本 1.4.x

- Windows Server 2019 STIG 版本 2 发行版 2
- Windows Server 2016 STIG 版本 2 发行版 2
- Windows Server 2012 R2 MS STIG 版本 3 发行版 2
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 1 发行版 7
- Internet Explorer 11 STIG 版本 1 发行版 19

STIG-Build-Windows-Medium 版本 1.4.x

- Windows Server 2019 STIG 版本 2 发行版 2
- Windows Server 2016 STIG 版本 2 发行版 2
- Windows Server 2012 R2 MS STIG 版本 3 发行版 2
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 1 发行版 7
- Internet Explorer 11 STIG 版本 1 发行版 19

STIG-Build-Windows-High 版本 1.4.x

- Windows Server 2019 STIG 版本 2 发行版 2
- Windows Server 2016 STIG 版本 2 发行版 2
- Windows Server 2012 R2 MS STIG 版本 3 发行版 2
- Microsoft .NET Framework 4.0 STIG 版本 2 发行版 1
- Windows Firewall STIG 版本 1 发行版 7
- Internet Explorer 11 STIG 版本 1 发行版 19

Linux STIG 强化组件

本节包含有关 Linux STIG 强化组件的信息，后面是版本历史记录日志。如果 Linux 发行版没有自己的 STIG 设置，则强化组件会应用 RHEL 设置。

Linux 组件具有可选的输入参数，可帮助您自定义 Linux 实例的以下行为。

- 级别（字符串）如果未指定任何值，则默认值为High并将应用所有适用的低、中和高配置。
- InstallPackages （字符串）如果值为No，则该组件不安装任何其他软件包。如果该值为Yes，则该组件将安装实现最大合规性所需的其他软件包。默认值为 No。
- SetDoDConsent横幅（字符串）如果值为No，则当您连接到安装了 STIG Linux 组件之一的实例时，不会显示国防部同意横幅。如果该值为Yes，则当您连接到安装了 STIG Linux 组件之一的实例时，会在您登录之前显示国防部同意横幅。您必须先确认该横幅，然后才能登录。默认值为 No。

  有关同意横幅的示例，请参阅当您访问 DLA 文档服务网站时显示的 Disclaimer Department of Defense Privacy and Consent Notice。

强化组件将支持的 STIG 设置应用于基于 Linux 发行版的基础架构，如下所示：

Red Hat Enterprise Linux (RHEL) 7 STIG 设置

- RHEL 7
- CentOS 7
- 亚马逊 Linux (2AL2)

RHEL 8 STIG 设置

- RHEL 8
- CentOS 8

RHEL 9 STIG 设置

- RHEL 9
- CentOS Stream 9

Linux STIG 低（第三类）

以下列表包含强化组件适用于您的基础架构的 STIG 设置。如果支持的设置不适用于您的基础架构，则强化组件会跳过该设置并继续向前。例如，某些 STIG 设置可能不适用于独立服务器。组织特定的策略也可能影响强化组件适用于哪些设置，如针对管理员查看文档设置的要求。

有关完整列表，请参阅STIGs 文档库。有关如何查看完整列表的信息，请参阅 STIG 查看工具。

RHEL 7 STIG 版本 3 发行版 15

- RHEL 7/CentOS 7/ AL2

  V-204452、V-204576 和 V-204605

RHEL 8 STIG 第 2 版 6

- RHEL 8/CentOS 8

  V-230241、V-230269、V-230270、V-230281、V-230285、V-230346、V-230381、V-230395、V-230468 和 V-244527

RHEL 9 STIG 第 2 版 7

- RHEL 9/CentOS Stream 9

  V-257782、V-257824、V-258138、V-258037、V-257880、V-258069、V-258076 V-258067 V-257946 V-257947 V-257795 V-257796 V-258173

亚马逊 Linux 2023 STIG 第 1 版 2

V-274141

SLES 12 STIG 版本 3 版本 4

V-217108、V-217113、V-217140、V-217198、V-217209、V-217211、V-217212、V-217213、V-217214、和 V-255915

SLES 15 STIG 版本 2 版本 6

V-234811、V-234850、V-234868、V-234873、V-234905、V-234907、V-234908 V-234909 V-234933 V-234934 V-234935 V-234936 V-234955 V-234963 V-234967 V-255921

Ubuntu 18.04 STIG 版本 2 版本 15

V-219163、V-219164、V-219165、V-219172、V-219173、V-219174、V-219175、V-219178、V-219179、和 V-219333

Ubuntu 20.04 STIG 版本 2 版本 4

V-238202、V-238203、V-238221、V-238222、V-238223、V-238224、V-238226、V-238234、V-238235、和 V-238373

Ubuntu 22.04 STIG 版本 2 第 7 版

V-260472、V-260476、V-260479、V-260480、V-260481、V-260520、V-260521、V-260549、V-260550、和 V-260596

Ubuntu 24.04 STIG 第 1 版 4

V-270645、V-270646、V-270664、V-270677、V-270690、V-270695、V-270706 V-270710 V-270734 V-270749 V-270752 V-270818 V-270820

Linux STIG 中等（第二类）

以下列表包含强化组件适用于您的基础架构的 STIG 设置。如果支持的设置不适用于您的基础架构，则强化组件会跳过该设置并继续向前。例如，某些 STIG 设置可能不适用于独立服务器。组织特定的策略也可能影响强化组件适用于哪些设置，如针对管理员查看文档设置的要求。

有关完整列表，请参阅STIGs 文档库。有关如何查看完整列表的信息，请参阅 STIG 查看工具。

> **Note**
>
> Linux STIG Medium 强化组件包括所有列出的 AWSTOE 适用于 Linux STIG Low 强化组件的 STIG 设置，以及专门针对第二类漏洞列出的 STIG 设置。

RHEL 7 STIG 版本 3 发行版 15

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

- RHEL 7/CentOS 7/ AL2

  V-204405、V-204406、V-204407、V-204408、V-204409、V-204410、V-204411、V-204412、V-204413
  4514、V-204515、V-204516、V-204517、V-204521、V-204524、V-204531、V-204536、V-204537、V-
  4548，V-204549，V-204550、V-204551、V-204552、V-204553、V-204554、V-204556、V-204557、V-
  4610、V-204611、V-204612、V-204613、V-204614、V-204615、V-204616、V-204617、V-204622、V-
  和 V-256970

RHEL 8 STIG 第 2 版 6

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

- RHEL 8/CentOS 8

V-230222、V-230228、V-230231、V-230233、V-230236、V-230237、V-230238、V-230239、V-230240
30277、V-230278、V-230279、V-230280、V-230282、V-230282、V-230286、V-230287、V-230288、V
13、V-230314、V-230315、V-230316、V-230318、V-230319、V-230320、V-230321、V-230321、V-230
V-230446、V-230447、V-230448、V-230449、V-230455、V-230456、V-230462、V-230463、V-230464
473、V-230474、V-230475、V-230478、V-230480、V-230481、V-230482、V-230483、V-230483、V-23
43、V-244544、V-244545、V-244547、V-244550、V-244551、V-244552、V-244552，V-244553、V-244
V-250317 V-251707 V-251708 V-251709 V-251710 V-251711 V-251713 V-251714 V-251715
V-251716 V-251717 V-251718 V-256974

RHEL 9 STIG 第 2 版 7

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

- RHEL 9/CentOS Stream 9

V-257780、V-257781、V-257786、V-257786、V-257788、V-257791、V-257792、V-257793、V-257794
8、V-257909、V-257910、V-257911、V-257912、V-257913、V-257914、V-257915、V-257916、V-257
7927、V-257928、V-257929、V-257930、V-257933、V-257934、V-257935、V-257939、V-257940、V-
7958，V-257959，V-257960、V-257961、V-257962、V-257963、V-257964、V-257966、V-257967、V-
V-258201 V-258202 V-258203 V-258204
222、V-258223、V-258224、V-258226、V-258226、V-258227、V-258228、V-258229、V-258232、V-25
175、V-270176、V-270177、V-272488 和 V-279936 V-258205 V-258206 V-258207 V-258208
V-258209 V-258210 V-258211 V-258212 V-258213

亚马逊 Linux 2023 STIG 第 1 版 2

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-273995、V-274000、V-274001、V-274002、V-274003、V-274004、V-274005、V-274006、V-274008、
V-274144 V-274145 V-274147 V-274149 -274177、V-274181、V-274182、V-274185 和 V-274187
V-274151 V-274152 V-274154 V-274155 V-274156 V-274157 V-274160 V-274161 V-274162

SLES 12 STIG 版本 3 版本 4

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-217102、V-217105、V-217105、V-217106、V-217106、V-217116、V-217117、V-217118、V-217119、
17253、V-217254、V-217255、V-217257、V-217258、V-217260、V-217265、V-217266、V-217267、V-2

278、V-217279、V-217280、V-217283、V-217284、V-217286、V-217287、V-217288、V-217289、V-217
V-237615 V-237616 V-237617 V-237618 V-237619 V-237620 V-237621 V-237622 V-237623
V-251720 V-251722 V-255914 V-256981

SLES 15 STIG 版本 2 版本 6

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-234802、V-234807、V-234808、V-234809、V-234813、V-234815、V-234817、V-234821、V-234822、
96、V-234897、V-234899，V-234900、V-234901、V-234902、V-234903、V-234904、V-2349906、V-234
V-235020 V-235021 V-235022 V-235023 V-235024 V-235025 V-235026 V-235028 V-235029
V-235030 V-251724 V-255920 V-256983

Ubuntu 18.04 STIG 版本 2 版本 15

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-219149、V-219155、V-219156、V-219160、V-219166、V-219168、V-219176、V-219181、V-219184、
和 V-255906

Ubuntu 20.04 STIG 版本 2 版本 4

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-238200、V-238205、V-238207、V-238209、V-238210、V-238211、V-238212、V-238213、V-238220、
和 V-274853

Ubuntu 22.04 STIG 版本 2 第 7 版

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-260471、V-260473、V-260474、V-260475、V-260475、V-260477、V-260478、V-260486、V-260487、
25、V-260626、V-260627、V-260628、V-260629、V-260630、V-260631、V-260632、V-260633、V-2606
V-260642 V-260643 V-260644 V-260645 V-260646 V-260647 V-260648 V-260649
V-274862、、V-274864 V-274866

Ubuntu 24.04 STIG 第 1 版 4

包括强化组件适用于此 Linux 发行版的 III 类（低级）漏洞的所有支持的 STIG 设置，以及：

V-270649、V-270651、V-270652、V-270653、V-270654、V-270656、V-270657、V-270659、V-270660、
和 V-274873

Linux STIG 高（第一类）

以下列表包含强化组件适用于您的基础架构的 STIG 设置。如果支持的设置不适用于您的基础架构，则强化组件会跳过该设置并继续向前。例如，某些 STIG 设置可能不适用于独立服务器。组织特定的策略也可能影响强化组件适用于哪些设置，如针对管理员查看文档设置的要求。

有关完整列表，请参阅STIGs 文档库。有关如何查看完整列表的信息，请参阅 STIG 查看工具。

> **Note**
>
> Linux STIG High 强化组件包括所有列出的 AWSTOE 适用于 Linux STIG Low 和 Linux STIG Medium 强化组件的 STIG 设置，以及列出的专门适用于 I 类漏洞的 STIG 设置。

RHEL 7 STIG 版本 3 发行版 15

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

- RHEL 7/CentOS 7/ AL2

  V-204424、V-204425、V-204442、V-204443、V-204447、V-204448、V-204455 V-204462 V-204497 V-204497 V-204502 V-204594 V-204620 V-204621

RHEL 8 STIG 第 2 版 6

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

- RHEL 8/CentOS 8

  V-230223、V-230264、V-230283、V-230284、V-230487、V-230492、V-230533 V-230558 V-244540 V-279933 V-230265 V-230226 V-230530 V-268322 V-230529 V-230531

RHEL 9 STIG 第 2 版 7

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

- RHEL 9/CentOS Stream 9

V-257820、V-257821、V-257826、V-257835、V-257955、V-257956、V-258059 V-258230 V-258238 V-257984 V-257986 V-258078 V-258094 V-258235 V-257784 V-257785

亚马逊 Linux 2023 STIG 第 1 版 2

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-273996、V-273997、V-273999、V-274007、V-274038、V-274039、V-274046、V-274052、V-274057 和 V-274153

SLES 12 STIG 版本 3 版本 4

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-217101、V-217139、V-217141、V-217142、V-217159、V-217160、V-217164 V-217264 V-217268 V-222386 V-251721

SLES 15 STIG 版本 2 版本 6

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-234800、V-234804、V-234818、V-234852、V-234859、V-234860、V-234898、V-234984、V-234985、和 V-251725

Ubuntu 18.04 STIG 版本 2 版本 15

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-219157、V-219158、V-219177、V-219212、V-219308、V-219314、V-219316 和 V-251507

Ubuntu 20.04 STIG 版本 2 版本 4

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-238201、V-238218、V-238219、V-238326、V-238327、V-238380 和 V-251504

Ubuntu 22.04 STIG 版本 2 第 7 版

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-260469、V-260482、V-260483、V-260523、V-260524、V-260526、V-260529 V-260539 V-260570 V-260571 V-260579 V-279937

Ubuntu 24.04 STIG 第 1 版 4

包括强化组件适用于此 Linux 发行版的类别 II 和 III（中级和低级）漏洞的所有支持的 STIG 设置，以及：：

V-270647、V-270648、V-270665、V-270666、V-270708、V-270711、V-270712 V-270713 V-270714 V-270717 V-270736 V-270738 V-279938

适用于 Linux 的 STIG 版本历史记录日志

本节记录 Linux 组件的版本历史记录。要查看一个季度的更改和已发布的版本，请选择标题以展开信息。

2026 年第一季度变化——2026 年 3 月 10 日：

更新了以下 STIG 版本，这些版本适用 STIGs 于 2026 年第一季度版本，适用于所有合规级别：

Stig-build-Linu

- RHEL 8 STIG 第 2 版 6
- RHEL 9 STIG 第 2 版 7
- 亚马逊 Linux 2023 STIG 第 1 版 2
- SLES 12 STIG 版本 3 版本 4
- SLES 15 STIG 版本 2 版本 6
- Ubuntu 20.04 STIG 版本 2 版本 4
- Ubuntu 22.04 STIG 版本 2 第 7 版
- Ubuntu 24.04 STIG 第 1 版 4

2025 年第三季度更改 - 2025 年 9 月 4 日：

添加了对 SUSE Linux Enterprise Server（SLES）操作系统和 Amazon Linux 2023 的支持。更新了以下 STIG 版本，并在 2025 年第三季度针对所有合规级别应用了 STIG 版本 ()low/medium/high：

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 4
- RHEL 9 STIG 版本 2 发行版 5
- Amazon Linux 2023 STIG 版本 1 发行版 1
- SLES 12 STIG 版本 3 发行版 3
- SLES 15 STIG 版本 2 发行版 5
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 3
- Ubuntu 22.04 STIG 版本 2 发行版 5
- Ubuntu 24.04 STIG 版本 1 发行版 2

2025 年第二季度更改 - 2025 年 6 月 26 日：

更新了以下 STIG 版本，并对 2025 年第二季度发行版应用了 STIG：

STIG-Build-Linux-Low 版本 2025.2.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 3
- RHEL 9 STIG 版本 2 发行版 4
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 2
- Ubuntu 22.04 STIG 版本 2 发行版 4
- Ubuntu 24.04 STIG 版本 1 发行版 1

STIG-Build-Linux-Medium 版本 2025.2.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 3
- RHEL 9 STIG 版本 2 发行版 4
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 2
- Ubuntu 22.04 STIG 版本 2 发行版 4

- Ubuntu 24.04 STIG 版本 1 发行版 1

STIG-Build-Linux-High 版本 2025.2.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 3
- RHEL 9 STIG 版本 2 发行版 4
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 2
- Ubuntu 22.04 STIG 版本 2 发行版 4
- Ubuntu 24.04 STIG 版本 1 发行版 1

2025 年第一季度更改 - 2025 年 4 月 11 日：

更新了以下 STIG 版本，对 2025 年第一季度发行版应用了 STIG，并添加了对 Ubuntu 24.04 的支持：

STIG-Build-Linux-Low 版本 2025.1.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 2
- RHEL 9 STIG 版本 2 发行版 3
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 2
- Ubuntu 22.04 STIG 版本 2 发行版 3
- Ubuntu 24.04 STIG 版本 1 发行版 1

STIG-Build-Linux-Medium 版本 2025.1.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 2
- RHEL 9 STIG 版本 2 发行版 3
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 2
- Ubuntu 22.04 STIG 版本 2 发行版 3

- Ubuntu 24.04 STIG 版本 1 发行版 1

STIG-Build-Linux-High 版本 2025.1.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 2
- RHEL 9 STIG 版本 2 发行版 3
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 2
- Ubuntu 22.04 STIG 版本 2 发行版 3
- Ubuntu 24.04 STIG 版本 1 发行版 1

2024 年第四季度更改 - 2024 年 12 月 10 日：

更新了以下 STIG 版本，对 2024 年第四季度发行版应用了 STIG，并添加了有关 Linux 组件的两个新输入参数的信息：

STIG-Build-Linux-Low 版本 2024.4.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 1
- RHEL 9 STIG 版本 2 发行版 2
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 1
- Ubuntu 22.04 STIG 版本 2 发行版 2

STIG-Build-Linux-Medium 版本 2024.4.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 1
- RHEL 9 STIG 版本 2 发行版 2
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 1
- Ubuntu 22.04 STIG 版本 2 发行版 2

STIG-Build-Linux-High 版本 2024.4.x

- RHEL 7 STIG 版本 3 发行版 15
- RHEL 8 STIG 版本 2 发行版 1
- RHEL 9 STIG 版本 2 发行版 2
- Ubuntu 18.04 STIG 版本 2 发行版 15
- Ubuntu 20.04 STIG 版本 2 发行版 1
- Ubuntu 22.04 STIG 版本 2 发行版 2

2024 年第三季度更改 - 2024 年 10 月 4 日（无更改）：

2024 年第三季度发行版的 Linux 组件 STIG 无更改。

2024 年第二季度更改 - 2024 年 5 月 10 日：

更新了 STIG 版本，并对 2024 年第二季度发行版应用了 STIG。还新增了对 RHEL 9、CentOS Stream 9 和 Ubuntu 22.04 的支持，如下所示：

STIG-Build-Linux-Low 版本 2024.2.x

- RHEL 7 STIG 版本 3 发行版 14
- RHEL 8 STIG 版本 1 发行版 14
- RHEL 9 STIG 版本 1 发行版 3
- Ubuntu 18.04 STIG 版本 2 发行版 14
- Ubuntu 20.04 STIG 版本 1 发行版 12
- Ubuntu 22.04 STIG 版本 1 发行版 1

STIG-Build-Linux-Medium 版本 2024.2.x

- RHEL 7 STIG 版本 3 发行版 14
- RHEL 8 STIG 版本 1 发行版 14
- RHEL 9 STIG 版本 1 发行版 3
- Ubuntu 18.04 STIG 版本 2 发行版 14
- Ubuntu 20.04 STIG 版本 1 发行版 12

- Ubuntu 22.04 STIG 版本 1 发行版 1

STIG-Build-Linux-High 版本 2024.2.x

- RHEL 7 STIG 版本 3 发行版 14
- RHEL 8 STIG 版本 1 发行版 14
- RHEL 9 STIG 版本 1 发行版 3
- Ubuntu 18.04 STIG 版本 2 发行版 14
- Ubuntu 20.04 STIG 版本 1 发行版 12
- Ubuntu 22.04 STIG 版本 1 发行版 1

2024 年第一季度更改 - 2024 年 2 月 6 日：

更新了 STIG 版本，并对 2024 年第一季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 2024.1.x

- RHEL 7 STIG 版本 3 发行版 14
- RHEL 8 STIG 版本 1 发行版 13
- Ubuntu 18.04 STIG 版本 2 发行版 13
- Ubuntu 20.04 STIG 版本 1 发行版 11

STIG-Build-Linux-Medium 版本 2024.1.x

- RHEL 7 STIG 版本 3 发行版 14
- RHEL 8 STIG 版本 1 发行版 13
- Ubuntu 18.04 STIG 版本 2 发行版 13
- Ubuntu 20.04 STIG 版本 1 发行版 11

STIG-Build-Linux-High 版本 2024.1.x

- RHEL 7 STIG 版本 3 发行版 14
- RHEL 8 STIG 版本 1 发行版 13
- Ubuntu 18.04 STIG 版本 2 发行版 13
- Ubuntu 20.04 STIG 版本 1 发行版 11

2023 年第四季度更改 - 2023 年 12 月 7 日：

更新了 STIG 版本，并对 2023 年第四季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 2023.4.x

- RHEL 7 STIG 版本 3 发行版 13
- RHEL 8 STIG 版本 1 发行版 12
- Ubuntu 18.04 STIG 版本 2 发行版 12
- Ubuntu 20.04 STIG 版本 1 发行版 10

STIG-Build-Linux-Medium 版本 2023.4.x

- RHEL 7 STIG 版本 3 发行版 13
- RHEL 8 STIG 版本 1 发行版 12
- Ubuntu 18.04 STIG 版本 2 发行版 12
- Ubuntu 20.04 STIG 版本 1 发行版 10

STIG-Build-Linux-High 版本 2023.4.x

- RHEL 7 STIG 版本 3 发行版 13
- RHEL 8 STIG 版本 1 发行版 12
- Ubuntu 18.04 STIG 版本 2 发行版 12
- Ubuntu 20.04 STIG 版本 1 发行版 10

2023 年第三季度更改 — 2023 年 10 月 4 日：

更新了 STIG 版本，并对 2023 年第三季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 2023.3.x

- RHEL 7 STIG 版本 3 发行版 12
- RHEL 8 STIG 版本 1 发行版 11
- Ubuntu 18.04 STIG 版本 2 发行版 11
- Ubuntu 20.04 STIG 版本 1 发行版 9

STIG-Build-Linux-Medium 版本 2023.3.x

- RHEL 7 STIG 版本 3 发行版 12
- RHEL 8 STIG 版本 1 发行版 11
- Ubuntu 18.04 STIG 版本 2 发行版 11
- Ubuntu 20.04 STIG 版本 1 发行版 9

STIG-Build-Linux-High 版本 2023.3.x

- RHEL 7 STIG 版本 3 发行版 12
- RHEL 8 STIG 版本 1 发行版 11
- Ubuntu 18.04 STIG 版本 2 发行版 11
- Ubuntu 20.04 STIG 版本 1 发行版 9

2023 年第二季度更改 — 2023 年 5 月 3 日：

更新了 STIG 版本，并对 2023 年第二季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 2023.2.x

- RHEL 7 STIG 版本 3 发行版 11
- RHEL 8 STIG 版本 1 发行版 10
- Ubuntu 18.04 STIG 版本 2 发行版 11
- Ubuntu 20.04 STIG 版本 1 发行版 8

STIG-Build-Linux-Medium 版本 2023.2.x

- RHEL 7 STIG 版本 3 发行版 11
- RHEL 8 STIG 版本 1 发行版 10
- Ubuntu 18.04 STIG 版本 2 发行版 11
- Ubuntu 20.04 STIG 版本 1 发行版 8

STIG-Build-Linux-High 版本 2023.2.x

- RHEL 7 STIG 版本 3 发行版 11

- RHEL 8 STIG 版本 1 发行版 10
- Ubuntu 18.04 STIG 版本 2 发行版 11
- Ubuntu 20.04 STIG 版本 1 发行版 8

2023 年第一季度更改 — 2023 年 3 月 27 日：

更新了 STIG 版本，并对 2023 年第一季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 2023.1.x

- RHEL 7 STIG 版本 3 发行版 10
- RHEL 8 STIG 版本 1 发行版 9
- Ubuntu 18.04 STIG 版本 2 发行版 10
- Ubuntu 20.04 STIG 版本 1 发行版 7

STIG-Build-Linux-Medium 版本 2023.1.x

- RHEL 7 STIG 版本 3 发行版 10
- RHEL 8 STIG 版本 1 发行版 9
- Ubuntu 18.04 STIG 版本 2 发行版 10
- Ubuntu 20.04 STIG 版本 1 发行版 7

STIG-Build-Linux-High 版本 2023.1.x

- RHEL 7 STIG 版本 3 发行版 10
- RHEL 8 STIG 版本 1 发行版 9
- Ubuntu 18.04 STIG 版本 2 发行版 10
- Ubuntu 20.04 STIG 版本 1 发行版 7

2022 年第四季度更改 — 2023 年 2 月 1 日：

更新了 STIG 版本，并对 2022 年第四季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 2022.4.x

- RHEL 7 STIG 版本 3 发行版 9

- RHEL 8 STIG 版本 1 发行版 8
- Ubuntu 18.04 STIG 版本 2 发行版 9
- Ubuntu 20.04 STIG 版本 1 发行版 6

STIG-Build-Linux-Medium 版本 2022.4.x

- RHEL 7 STIG 版本 3 发行版 9
- RHEL 8 STIG 版本 1 发行版 8
- Ubuntu 18.04 STIG 版本 2 发行版 9
- Ubuntu 20.04 STIG 版本 1 发行版 6

STIG-Build-Linux-High 版本 2022.4.x

- RHEL 7 STIG 版本 3 发行版 9
- RHEL 8 STIG 版本 1 发行版 8
- Ubuntu 18.04 STIG 版本 2 发行版 9
- Ubuntu 20.04 STIG 版本 1 发行版 6

2022 年第三季度更改 — 2022 年 9 月 30 日（无更改）：

2022 年第三季度发行版的 Linux 组件 STIGS 无更改。

2022 年第二季度更改 — 2022 年 8 月 2 日：

引入了 Ubuntu 支持，更新了 STIG 版本，并对 2022 年第二季度发行版应用了 STIGS，如下所示：

STIG-Build-Linux-Low 版本 2022.2.x

- RHEL 7 STIG 版本 3 发行版 7
- RHEL 8 STIG 版本 1 发行版 6
- Ubuntu 18.04 STIG 版本 2 发行版 6（全新）
- Ubuntu 20.04 STIG 版本 1 发行版 4（全新）

STIG-Build-Linux-Medium 版本 2022.2.x

- RHEL 7 STIG 版本 3 发行版 7

- RHEL 8 STIG 版本 1 发行版 6
- Ubuntu 18.04 STIG 版本 2 发行版 6（全新）
- Ubuntu 20.04 STIG 版本 1 发行版 4（全新）

STIG-Build-Linux-High 版本 2022.2.x

- RHEL 7 STIG 版本 3 发行版 7
- RHEL 8 STIG 版本 1 发行版 6
- Ubuntu 18.04 STIG 版本 2 发行版 6（全新）
- Ubuntu 20.04 STIG 版本 1 发行版 4（全新）

2022 年第一季度更改 — 2022 年 4 月 26 日：

已重构以包括对容器更好的支持。将之前的 AL2 脚本与 RHEL 7 结合使用。更新了 STIG 版本，并对 2022 年第一季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 3.6.x

- RHEL 7 STIG 版本 3 发行版 6
- RHEL 8 STIG 版本 1 发行版 5

STIG-Build-Linux-Medium 版本 3.6.x

- RHEL 7 STIG 版本 3 发行版 6
- RHEL 8 STIG 版本 1 发行版 5

STIG-Build-Linux-High 版本 3.6.x

- RHEL 7 STIG 版本 3 发行版 6
- RHEL 8 STIG 版本 1 发行版 5

2021 年第四季度更改 — 2021 年 12 月 20 日：

更新了 STIG 版本，并对 2021 年第四季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 3.5.x

- RHEL 7 STIG 版本 3 发行版 5
- RHEL 8 STIG 版本 1 发行版 4

STIG-Build-Linux-Medium 版本 3.5.x

- RHEL 7 STIG 版本 3 发行版 5
- RHEL 8 STIG 版本 1 发行版 4

STIG-Build-Linux-High 版本 3.5.x

- RHEL 7 STIG 版本 3 发行版 5
- RHEL 8 STIG 版本 1 发行版 4

2021 年第三季度更改 — 2021 年 9 月 30 日：

更新了 STIG 版本，并对 2021 年第三季度发行版应用了 STIG，如下所示：

STIG-Build-Linux-Low 版本 3.4.x

- RHEL 7 STIG 版本 3 发行版 4
- RHEL 8 STIG 版本 1 发行版 3

STIG-Build-Linux-Medium 版本 3.4.x

- RHEL 7 STIG 版本 3 发行版 4
- RHEL 8 STIG 版本 1 发行版 3

STIG-Build-Linux-High 版本 3.4.x

- RHEL 7 STIG 版本 3 发行版 4
- RHEL 8 STIG 版本 1 发行版 3

SCAP 合规性验证器组件

安全内容自动化协议 (SCAP) 是一套标准，供 IT 专业人员用来识别应用程序安全漏洞以确保合规性。SCAP 合规性检查器 (SCC) 是一款经过 SCAP 验证的扫描工具，由大西洋海军信息战中心 (NIWC) 发布。有关更多信息，请参阅大西洋 NIWC 网站上的安全内容自动化协议 (SCAP) 合规性检查器 (SCC)。

AWSTOE `scap-compliance-checker-windows`和`scap-compliance-checker-linux`组件在管道构建和测试实例上下载并安装 SCC 扫描器。扫描器运行时，它会使用 DISA SCAP 基准测试执行经过身份验证的配置扫描，并提供包含以下信息的报告。 AWSTOE 还会将信息写入您的应用程序日志。

- 应用于实例的 STIG 设置。
- 实例的总体合规性分数。

我们建议您将运行 SCAP 验证作为构建过程的最后一步，以确保报告准确的合规性验证结果。

> **Note**
>
> 您可以使用 STIG 查看工具之一查看报告。这些工具可通过 DoD Cyber Exchange 在线获得。

以下各节介绍 SCAP 验证组件包含的基准。

scap-compliance-checker-windows 版本 2024.03.0

该`scap-compliance-checker-windows`组件在 Image Builder 为构建和测试映像而创建的 EC2 实例上运行。 AWSTOE 记录 SCC 应用程序生成的报告和分数。

该组件执行以下工作流程步骤：

1. 下载并安装 SCC 应用程序。
2. 导入合规性基准。
3. 使用 SCC 应用程序运行验证。
4. 将合规性报告和分数保存在本地构建实例桌面上。
5. 将合规性分数从本地报告记录到 AWSTOE 应用程序日志文件中。

> **Note**
>
> AWSTOE 目前支持 Windows Server 2012 R2 MS、2016、2019 和 2022 年的 SCAP 合规性验证。

适用于 Windows 的 SCAP 合规性检查器组件包括以下基准：

SCC 版本：5.10

2023 年第四季度基准：

- u_ms_defender_antivirus_v2r5_stig_scap_1-2_Benchmark
- U_MS_ _framework_4-0_v2r2_stig_scap_1-2_Benchmark DotNet
- U_MS_ _v2r6_stig_scap_1-2_Benchmark IE11
- u_ms_windows_2012_and_2012_r2_dc_v3r5_stig_scap_1-2_Benchmark
- u_ms_windows_2012_and_2012_r2_ms_v3r5_stig_scap_1-2_Benchmark
- u_ms_windows_defender_firewall_v2r3_stig_scap_1-2_Benchmark
- u_ms_windows_server_2016_v2r7_stig_scap_1-2_Benchmark
- u_ms_Windows_server_2019_v3r2_stig_scap_1-2_Benchmark
- u_ms_windows_server_2022_v2r2_stig_scap_1-2_Benchmark
- u_can_ubuntu_20-04_lts_v1r10_stig_scap_1-2_Benchmark
- u_rhel_7_v3r15_stig_scap_1-3_Benchmark
- u_rhel_8_v1r13_stig_scap_1-3_Benchmark
- u_rhel_9_v2r1_stig_scap_1-3_Benchmark

scap-compliance-checker-linux 版本 2021.04.0

该`scap-compliance-checker-linux`组件在 Image Builder 为构建和测试映像而创建的 EC2 实例上运行。 AWSTOE 记录 SCC 应用程序生成的报告和分数。

该组件执行以下工作流程步骤：

1. 下载并安装 SCC 应用程序。
2. 导入合规性基准。

3. 使用 SCC 应用程序运行验证。
4. 将合规性报告和分数保存在本地构建实例的以下位置：`/opt/scc/SCCResults`。
5. 将合规性分数从本地报告记录到 AWSTOE 应用程序日志文件中。

> **Note**
>
> AWSTOE 目前支持 RHEL 7/8 和 Ubuntu 18.04/20.04 的 SCAP 合规性验证。SCC 应用程序目前支持 x86 架构进行验证。

适用于 Linux 的 SCAP 合规性检查器组件包括以下基准：

SCC 版本：5.10

2023 年第四季度基准：

- u_can_ubuntu_20-04_lts_v1r10_stig_scap_1-2_Benchmark
- u_rhel_7_v3r15_stig_scap_1-3_Benchmark
- u_rhel_8_v1r13_stig_scap_1-3_Benchmark
- u_rhel_9_v2r1_stig_scap_1-3_Benchmark
- u_ms_defender_antivirus_v2r5_stig_scap_1-2_Benchmark
- U_MS_ _framework_4-0_v2r2_stig_scap_1-2_Benchmark DotNet
- U_MS_ _v2r6_stig_scap_1-2_Benchmark IE11
- u_ms_windows_2012_and_2012_r2_dc_v3r5_stig_scap_1-2_Benchmark
- u_ms_windows_2012_and_2012_r2_ms_v3r5_stig_scap_1-2_Benchmark
- u_ms_windows_defender_firewall_v2r3_stig_scap_1-2_Benchmark
- u_ms_windows_server_2016_v2r7_stig_scap_1-2_Benchmark
- u_ms_Windows_server_2019_v3r2_stig_scap_1-2_Benchmark
- u_ms_windows_server_2022_v2r2_stig_scap_1-2_Benchmark

SCAP 版本历史记录

下表列示了本文档中描述的 SCAP 环境和设置的重要更改。

更改	描述	日期
2025 年第一季度 SCAP 最新动态	• scap-compliance-checker-windows 版本 2024.03.0（SCC 版本：5.10） • scap-compliance-checker-windows 版本 2025.01.0（SCC 版本：5.10）	2025 年 4 月 11 日
2023 年第四季度 SCAP 更新	• scap-compliance-checker-windows 版本 2023.04.0（SCC 版本：5.8） • scap-compliance-checker-windows 版本 2023.04.0（SCC 版本：5.8）	2021 年 12 月 20 日
2023 年第三季度 SCAP 更新	• scap-compliance-checker-windows 版本 2023.03.0（SCC 版本：5.7.2） • scap-compliance-checker-linux 版本 2023.03.0 (SCC 版本：5.7.2)	2023 年 11 月 13 日
添加了 SCAP 组件	推出了以下 SCAP 组： • 已创建 scap-compliance-checker-linux版本 2021.04.0（SCC 版本：5.4.2） • 已创建 scap-compliance-checker-linux版本 2021.04.0（SCC 版本：5.4.2）	2021 年 12 月 20 日

为 Image Builder 映像开发自定义组件

您可以创建自己的组件，根据自己的确切规格自定义 Image Builder 映像。使用以下步骤为您的 Image Builder 映像或容器配方开发自定义组件。

1. 如果要开发组件文档并在本地对其进行验证，则可以安装 AWS Task Orchestrator and Executor (AWSTOE) 应用程序并在本地计算机上进行设置。有关更多信息，请参阅 手动设置以开发自定义组件 AWSTOE。
2. 创建使用组件文档框架的 AWSTOE 组件文档。有关文档框架的更多信息，请参阅使用 AWSTOE 组件文档框架创建自定义组件。
3. 创建自定义组件时，请指定您的组件文档。有关更多信息，请参阅 使用 Image Builder 创建一个自定义组件。

主题

在 Image Builder 中为自定义组件创建 YAML 组件文档

要构建组件，必须提供 YAML 或者 JSON 应用程序组件文档。该文档包含了在您为提供映像自定义而定义的阶段和步骤中运行的代码。

本节中的一些示例创建了一个构建组件，该组件在 AWSTOE 组件管理应用程序中调用`Update0S`操作模块。该模块更新操作系统。有关 `Update0S` 操作模块的更多信息，请参阅 UpdateOS。

macOS 操作系统示例使用 `ExecuteBash` 操作模块来安装和验证 `wget` 实用程序。`Update0S` 操作模块不支持 macOS。有关 `ExecuteBash` 操作模块的更多信息，请参阅 ExecuteBash。有关 AWSTOE 应用程序组件文档的阶段、步骤和语法的更多信息，请参阅在 AWSTOE中使用文档。

> **Note**
>
> Image Builder 根据组件文档中定义的阶段确定组件类型，如下所示：
>
> - 构建 - 这是默认的组件类型。任何未归类为测试组件的组件都是构建组件。这种类型的组件在映像构建阶段运行。如果此构建组件已定义 `test` 阶段，则该阶段在测试阶段运行。
> - 测试 - 要获得测试组件资格，组件文档必须仅包含一个名为 `test` 的阶段。对于与构建组件配置相关的测试，我们建议您不要使用独立的测试组件。相反，在关联的构建组件中使用 `test` 阶段。
>
> 有关 Image Builder 如何在其构建过程中使用阶段 (stage) 和时段 (phase) 来管理组件工作流的更多信息，请参阅 使用组件自定义 Image Builder 映像。

要为示例应用程序创建 YAML 应用程序组件文档，请按照与您的映像操作系统匹配的选项卡上的步骤进行操作。

Linux

创建 YAML 组件文件

使用文件编辑工具创建您的组件文档。文档示例使用名为 *update-linux-os.yaml* 的文件，包含如下内容：

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
name: update-linux-os
description: Updates Linux with the latest security updates.
schemaVersion: 1
phases:
  - name: build
    steps:
    - name: UpdateOS
      action: UpdateOS
# Document End
```

> **Tip**
>
> 在代码环境中使用像在线 YAML Validat 或 YAML lint 扩展这样的工具来验证 YAML 格式是否正确。

Windows

创建 YAML 组件文件

使用文件编辑工具创建您的组件文档。文档示例使用名为 *update-windows-os.yaml* 的文件，包含如下内容：

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
name: update-windows-os
description: Updates Windows with the latest security updates.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: UpdateOS
        action: UpdateOS
# Document End
```

> **Tip**
>
> 在代码环境中使用像在线 YAML Validat 或 YAML lint 扩展这样的工具来验证 YAML 格式是否正确。

macOS

创建 YAML 组件文件

使用文件编辑工具创建您的组件文档。文档示例使用名为 *wget-macos.yaml* 的文件，包含如下内容：

```
name: WgetInstallDocument
description: This is wget installation document.
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: WgetBuildStep
        action: ExecuteBash
        inputs:
          commands:
            - |
              PATH=/usr/local/bin:$PATH
              sudo -u ec2-user brew install wget


  - name: validate
    steps:
      - name: WgetValidateStep
        action: ExecuteBash
        inputs:
          commands:
            - |
              function error_exit {
                echo $1
                echo "{\"failureMessage\":\"$2\"}"
                exit 1
              }

              type wget
              if [ $? -ne 0 ]; then
                error_exit "$stderr" "Wget installation failed!"
              fi

  - name: test
    steps:
      - name: WgetTestStep
```

```
    action: ExecuteBash
    inputs:
      commands:
        - wget -h
```

> **Tip**
>
> 在代码环境中使用像在线 YAML Validat 或 YAML lint 扩展这样的工具来验证 YAML 格式是否正确。

使用 Image Builder 创建一个自定义组件

完成组件文档后，您可以使用它来创建 Image Builder 配方可以使用的自定义组件。您可以通过 Image Builder 控制台、API 或 SDKs命令行创建自定义组件。有关如何创建带输入参数的自定义组件并在配方中使用该组件的更多信息，请参阅教程：创建带有输入参数的自定义组件。

以下各节将向您介绍如何通过控制台或通过 AWS CLI创建组件。

内容

通过控制台创建自定义组件

要通过 Image Builder 控制台创建 AWSTOE 应用程序组件，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中选择组件。然后，选择 Create component (创建组件)。
3. 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：

    a. Image Operating system (OS) (镜像操作系统 (OS))。指定与该组件兼容的操作系统。

    b. 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。

c. Component name (组件名称)。输入组件的名称。

d. Component version (组件版本)。输入组件的版本号。

e. 描述。提供可选的描述以帮助您标识组件。

f. 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。

4. 在定义文档部分中，默认选项为定义文档内容。组件文档定义了 Image Builder 在构建和测试实例上执行的用于创建映像的操作。

   在内容框中，输入您的 YAML 组件文档内容。要从 Linux 的 Hello World 示例开始，请选择使用示例选项。要了解有关如何创建 YAML 组件文档或从该页面复制并粘贴 UpdateOS 示例的更多信息，请参阅 在 Image Builder 中为自定义组件创建 YAML 组件文档。

5. 在输入组件详细信息后，选择创建组件。

   > **Note**
   >
   > 要在创建或更新配方时查看您的新组件，请将我拥有的筛选条件应用于构建或测试组件列表。筛选条件位于组件列表顶部的搜索框旁边。

6. 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)。

更新组件

要创建新的组件版本，请遵循以下步骤：

1. 取决于您从哪个位置开始：

   - 从组件列表页面 — 选中组件名称旁边的复选框，然后从操作菜单中选择创建新版本。
   - 从组件详情页面 — 选择标题右上角的创建新版本按钮。

2. 当显示创建组件页面时，组件信息已使用当前值进行填充。按照创建组件步骤更新组件。这样可以确保在组件版本中输入唯一的语义版本。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

从中创建自定义组件 AWS CLI

在本节中，您将学习如何设置和使用中的 Image Builder 命令 AWS CLI 来创建 AWSTOE 应用程序组件，如下所示。

- 将您的 YAML 组件文档上传到可以从命令行引用的 S3 存储桶。
- 使用create-component命令创建 AWSTOE 应用程序组件。
- 使用 list-components 命令和名称筛选器列出组件版本，以查看已存在哪些版本。您可以使用输出来确定应使用哪个版本进行更新。

要根据输入 YAML 文档创建 AWSTOE 应用程序组件，请按照与您的映像操作系统平台相匹配的步骤进行操作。

Linux

将您的应用程序组件文档存储到 Amazon S3 中

您可以使用 S3 存储桶作为 AWSTOE 应用程序组件源文档的存储库。要存储组件文档，请按照以下步骤操作：

- 将文档上传到 Amazon S3

  如果您的文档小于 64 KB，则可以跳过此步骤。大小为 64 KB 或更大的文档必须存储在 Amazon S3 中。

  ```
  aws s3 cp update-linux-os.yaml s3://amzn-s3-demo-destination-bucket/my-path/update-linux-os.yaml
  ```

从 YAML 文档创建组件

要简化您在中使用的create-component命令 AWS CLI，请创建一个 JSON 文件，其中包含要传递给命令的所有组件参数。包括您在前面创建的 *update-linux-os.yaml* 文档的位置。`uri` 键值对包含文件引用。

> **Note**
>
> JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 命令请求参数，请参阅 EC2 Image Builder API 参考中的 CreateComponent 命令。要将数据值作为命令行参数提供，请参阅《AWS CLI 命令引用》中指定的参数名称。

1. 创建 CLI 输入 JSON 文件

   使用文件编辑工具创建名为 *create-update-linux-os-component.json* 的文件。包括以下内容：

```
{
 "name": "update-linux-os",
 "semanticVersion": "1.1.2",
 "description": "An example component that updates the Linux operating system",
 "changeDescription": "Initial version.",
 "platform": "Linux",
 "uri": "s3://amzn-s3-demo-destination-bucket/my-path/update-linux-os.yaml",
 "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
b123-456b-7f89-0123456f789c",
 "tags": {
  "MyTagKey-purpose": "security-updates"
 }
}
```

2. 创建组件

   使用以下命令创建组件，引用您在上一步中创建的 JSON 文件的文件名：

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

   > **Note**
   >
   > - JSON 文件路径开头必须包含 `file://` 符号。
   > - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

Windows

将您的应用程序组件文档存储到 Amazon S3 中

您可以使用 S3 存储桶作为 AWSTOE 应用程序组件源文档的存储库。要存储组件文档，请按照以下步骤操作：

- 将文档上传到 Amazon S3

  如果您的文档小于 64 KB，则可以跳过此步骤。大小为 64 KB 或更大的文档必须存储在 Amazon S3 中。

  ```
  aws s3 cp update-windows-os.yaml s3://amzn-s3-demo-destination-bucket/my-path/update-windows-os.yaml
  ```

从 YAML 文档创建组件

要简化您在中使用的create-component命令 AWS CLI，请创建一个 JSON 文件，其中包含要传递给命令的所有组件参数。包括您在前面创建的 *update-windows-os.yaml* 文档的位置。uri 键值对包含文件引用。

> **Note**
>
> JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 命令请求参数，请参阅 EC2 Image Builder API 参考中的 CreateComponent 命令。要将数据值作为命令行参数提供，请参阅《AWS CLI 命令引用》中指定的参数名称。

1. 创建 CLI 输入 JSON 文件

   使用文件编辑工具创建名为 *create-update-windows-os-component.json* 的文件。包括以下内容：

   ```
   {
    "name": "update-windows-os",
    "semanticVersion": "1.1.2",
    "description": "An example component that updates the Windows operating system.",
    "changeDescription": "Initial version.",
    "platform": "Windows",
    "uri": "s3://amzn-s3-demo-destination-bucket/my-path/update-windows-os.yaml",
    "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-b123-456b-7f89-0123456f789c",
    "tags": {
     "MyTagKey-purpose": "security-updates"
    }
   }
   ```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

2. 创建组件

   使用以下命令创建组件，引用您在上一步中创建的 JSON 文件的文件名：

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

   > **Note**
   >
   > - JSON 文件路径开头必须包含 `file://` 符号。
   > - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

macOS

将您的应用程序组件文档存储到 Amazon S3 中

您可以使用 S3 存储桶作为 AWSTOE 应用程序组件源文档的存储库。要存储组件文档，请按照以下步骤操作：

- 将文档上传到 Amazon S3

  如果您的文档小于 64 KB，则可以跳过此步骤。大小为 64 KB 或更大的文档必须存储在 Amazon S3 中。

  ```
  aws s3 cp wget-macos.yaml s3://amzn-s3-demo-destination-bucket/my-path/wget-macos.yaml
  ```

从 YAML 文档创建组件

要简化您在中使用的create-component命令 AWS CLI，请创建一个 JSON 文件，其中包含要传递给命令的所有组件参数。包括您在前面创建的 *wget-macos.yaml* 文档的位置。`uri` 键值对包含文件引用。

> **Note**
>
> JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 命令请求参数，请参阅 EC2 Image Builder API 参考中的 CreateComponent 命令。要将数据值作为命令行参数提供，请参阅《AWS CLI 命令引用》中指定的参数名称。

1. 创建 CLI 输入 JSON 文件

   使用文件编辑工具创建名为 *install-wget-macos-component.json* 的文件。包括以下内容：

   ```
   {
    "name": "install install-wget-macos-component",
    "semanticVersion": "1.1.2",
    "description": "An example component that installs and verifies the wget
    utility on macOS.",
    "changeDescription": "Initial version.",
    "platform": "macOS",
    "uri": "s3://amzn-s3-demo-destination-bucket/my-path/wget-macos.yaml",
    "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
   b123-456b-7f89-0123456f789c",
    "tags": {
     "MyTagKey-purpose": "install-software"
    }
   }
   ```

2. 创建组件

   使用以下命令创建组件，引用您在上一步中创建的 JSON 文件的文件名：

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

AWSTOE 组件版本控制，用于来自的更新 AWS CLI

AWSTOE 组件名称和版本嵌入在组件的 Amazon 资源名称 (ARN) 中，位于组件前缀之后。组件的每个新版本都有自己唯一的 ARN。创建新版本的步骤与创建新组件的步骤完全相同，前提是该组件名称的语义版本是唯一的。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

为确保分配下一个逻辑版本，请先获取要更改的组件的现有版本列表。使用带有 AWS CLI、并筛选名称的list-components命令。

在此示例中，您将根据在前面的 Linux 示例中创建的组件的名称进行筛选。要列出您创建的组件，请使用您在 create-component 命令中使用的 JSON 文件中的 `name` 参数值。

```
aws imagebuilder list-components --filters name="name",values="update-linux-os"
{
    "requestId": "123a4567-b890-123c-45d6-ef789ab0cd1e",
    "componentVersionList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:1234560087789012:component/update-
linux-os/1.0.0",
            "name": "update-linux-os",
            "version": "1.0.0",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2020-09-24T16:58:24.444Z"
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:1234560087789012:component/update-
linux-os/1.0.1",
            "name": "update-linux-os",
            "version": "1.0.1",
            "platform": "Linux",
```

```
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2021-07-10T03:38:46.091Z"
        }
    ]
}
```

根据您的结果，您可以确定下一个版本应该是什么。

导入脚本以从中创建组件 AWS CLI

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，您可以使用以下示例。

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，该文件直接引用了一个名为的 PowerShell 脚本AdminConfig.ps1，该脚本已上传到*amzn-s3-demo-source-bucket*。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://amzn-s3-demo-source-bucket/AdminConfig.ps1",
"kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/60763706-
b131-418b-8f85-3420912f020c"
}
```

要从导入后的脚本创建组件，请运行以下命令。

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

自动生成版本管理

当您创建与现有组件具有相同名称和语义版本的组件时，Image Builder 会自动增加构建版本（例如，从/1到/2/3、到，等等）。这使您无需手动管理版本号即可对组件进行迭代更新，这在 CI/CD 管道和 infrastructure-as-code部署中特别有用。如果组件内容与之前的构建版本相同，Image Builder 会返回ResourceAlreadyExistsException以防止重复的组件消耗您的服务配额。

使用版本引用

当您创建或检索组件时，Image Builder 会自动在对象中提供 ARNs 带有通配符的预构版本模式。latestVersionReferences 这些参考使您可以轻松地在配方和管道中使用最新版本的组件，而无需手动解析 ARNs。

选择正确的版本参考

- latestVersionArn (x.x.x)-始终使用绝对最新的组件版本。
- atestMajorVersionArn (1.x.x)-在主要版本中使用最新的次要版本和补丁版本。
- latestMinorVersionArn (1.2.x)-仅使用最新的补丁版本。
- latestPatchVersionArn (1.2.3)-引用特定的语义版本，但要获取最新的构建版本。

> **Note**
>
> 为避免意外费用，请务必清理根据本指南中的示例创建的资源和管道。有关删除 Image Builder 中的资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。

Image Builder 如何使用 AWS Task Orchestrator and Executor 应用程序管理组件

EC2 Image Builder 使用 AWS Task Orchestrator and Executor (AWSTOE) 应用程序来编排复杂的工作流程、修改系统配置和测试映像，而无需额外的 devops 脚本或代码。此应用程序管理和运行使用其声明性文档架构的组件。

AWSTOE 是一个独立的应用程序，当您创建映像时，Image Builder 会将其安装在其构建和测试实例上。您也可以将其手动安装在 EC2 实例上，以创建自己的自定义组件。它不需要任何其他设置，也可以在本地运行。

内容

- AWSTOE 下载
- 支持的区域
- AWSTOE 命令参考
- 手动设置以开发自定义组件 AWSTOE
- 使用 AWSTOE 组件文档框架创建自定义组件

- AWSTOE 组件管理器支持的操作模块
- 配置 AWSTOE 运行命令的输入

AWSTOE 下载

要安装 AWSTOE，请选择适用于您的架构和平台的下载链接。如果您连接到服务的 VPC 终端节点（例如 Image Builder），则该终端节点必须附加自定义终端节点策略，其中包括访问 S3 存储桶进行 AWSTOE 下载的权限。否则，您的构建和测试实例将无法下载引导脚本 (`bootstrap.sh`) 并安装 AWSTOE 应用程序。有关更多信息，请参阅 为 Image Builder 创建 VPC 端点策略。

> **Important**
>
> AWS 正在逐步取消对 TLS 版本 1.0 和 1.1 的支持。要访问 S3 存储桶进行 AWSTOE 下载，您的客户端软件必须使用 TLS 版本 1.2 或更高版本。有关更多信息，请参阅AWS 此博客文章。

架构	平台	下载链接	示例
386	AL 2 和 2023 RHEL 7、8 和 9 Ubuntu 16.04、18.04、20.04、22.04 和 24.04 CentOS 7 和 8 SUSE 12 和 15	`https://awstoe-<region>.s3.com/latest/linux/386/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/386/awstoe
AMD64	AL 2 和 2023 RHEL 7、8 和 9 Ubuntu 16.04、18.04、20.04、22.04 和 24.04 CentOS 7 和 8	`https://awstoe-<region>.s3.com/latest/linux/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/awstoe

架构	平台	下载链接	示例
	CentOS Stream 8 SUSE 12 和 15		
AMD64	macOS 10.14.x (Mojave)、10.15.x (Catalina)、11.x (Big Sur)、12.x (Monterey)	`https://awstoe-region.s3.rs.com/latest/darwin/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/darwin/amd64/awstoe
AMD64	Windows Server 2012 R2、2016、2019 和 2022	`https://awstoe-<region>.s3.com/latest/windows/amd64/awstoe.exe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/amd64/awstoe.exe
ARM64	AL 2 和 2023 RHEL 7、8 和 9 Ubuntu 16.04、18.04、20.04、22.04 和 24.04 CentOS 7 和 8 CentOS Stream 8 SUSE 12 和 15	`https://awstoe-<region>.s3.com/latest/linux/arm64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/arm64/awstoe

支持的区域

AWSTOE 在以下区域支持作为独立应用程序。

AWS 区域 名字	AWS 区域
美国东部（俄亥俄州）	us-east-2

AWS 区域 名字	AWS 区域
美国东部（弗吉尼亚州北部）	us-east-1
AWS GovCloud （美国东部）	us-gov-east-1
AWS GovCloud （美国西部）	us-gov-west-1
美国西部（加利福尼亚北部）	us-west-1
美国西部（俄勒冈州）	us-west-2
非洲（开普敦）	af-south-1
亚太地区（香港）	ap-east-1
亚太地区（大阪）	ap-northeast-3
亚太地区（首尔）	ap-northeast-2
亚太地区 (孟买)	ap-south-1
亚太地区（海得拉巴）	ap-south-2
亚太地区（新加坡）	ap-southeast-1
亚太地区（悉尼）	ap-southeast-2
亚太地区（雅加达）	ap-southeast-3
亚太地区（东京）	ap-northeast-1
加拿大（中部）	ca-central-1
欧洲地区（法兰克福）	eu-central-1
欧洲（苏黎世）	eu-central-2
欧洲地区（斯德哥尔摩）	eu-north-1
欧洲（米兰）	eu-south-1

AWS 区域 名字	AWS 区域
欧洲（西班牙）	eu-south-2
欧洲地区（爱尔兰）	eu-west-1
欧洲地区（伦敦）	eu-west-2
欧洲地区（巴黎）	eu-west-3
以色列（特拉维夫）	il-central-1
中东（阿联酋）	me-central-1
中东（巴林）	me-south-1
南美洲（圣保罗）	sa-east-1
中国（北京）	cn-north-1
中国（宁夏）	cn-northwest-1

AWSTOE 命令参考

AWSTOE 是一款在 Amazon EC2 实例上运行的命令行组件管理应用程序。当 Image Builder 启动 EC2 构建或测试实例时，它会安装在该实例 AWSTOE 上。然后，它会在中运行 AWSTOE 命令 AWS CLI 来安装或验证镜像或容器配方中指定的组件。

> **Note**
>
> 某些 AWSTOE 操作模块需要更高的权限才能在 Linux 服务器上运行。要使用提升的权限，请在命令语法前加上前缀 sudo，或者在登录时运行 sudo su 命令，然后再运行下面链接的命令。有关 AWSTOE 操作模块的更多信息，请参见AWSTOE 组件管理器支持的操作模块。

运行

使用 run 命令为一个或多个组件文档运行 YAML 文档脚本。

验证

使用 validate 命令为一个或多个组件文档验证 YAML 文档语法。

awstoe run command

该命令按 `--config` 参数指定的配置文件或 `--documents` 参数指定的组件文档列表中 YAML 组件文档脚本的内置顺序运行 YAML 组件文档脚本。

> **Note**
>
> 您必须准确指定下列参数之一，切勿同时指定两个参数：
> --config
> --documents

语法

```
awstoe run [--config <file path>] [--cw-ignore-failures <?>]
      [--cw-log-group <?>] [--cw-log-region us-west-2] [--cw-log-stream <?>]
      [--document-s3-bucket-owner <owner>] [--documents <file path,file path,...>]
      [--execution-id <?>] [--log-directory <file path>]
      [--log-s3-bucket-name <name>] [--log-s3-bucket-owner <owner>]
      [--log-s3-key-prefix <?>] [--parameters name1=value1,name2=value2...]
      [--phases <phase name>] [--state-directory <directory path>] [--version <?>]
      [--help] [--trace]
```

参数和选项

参数

config*./config-example.json*

简写形式：-c *./config-example.json*

配置文件（条件性）。此参数包含 JSON 文件的文件位置，该文件包含此命令正在运行的组件的配置设置。如果在配置文件中指定 run 命令设置，则不得指定 `--documents` 参数。有关输入配置的更多信息，请参阅 配置 AWSTOE 运行命令的输入。

有效位置包括：

- 本地文件路径 (*./config-example.json*)
- 一个 S3; URI。(s3://*bucket/key*)

--cw-ignore-failures

简写形式：不适用

忽略日志中的 CloudWatch 日志失败。

--cw-log-group

简写形式：不适用

CloudWatch 日志的LogGroup名称。

--cw-log-region

简写形式：不适用

适用于 CloudWatch 日志的 AWS 区域。

--cw-log-stream

简写形式：不适用

CloudWatch 日志的LogStream名称，用于指示将console.log文件传输到 AWSTOE 何处。

--document-s3-bucket-owner

简写形式：不适用

基于 S3 URI 的文档的存储桶拥有者账户 ID。

--文档 ***./doc-1.yaml,./doc-n.yaml***

简写形式：-d *./doc-1.yaml*,*./doc-n*

组件文档（条件性）。此参数包含以逗号分隔的文件位置列表，以供运行 YAML 组件文档。如果您使用 --documents 参数为 run 命令指定 YAML 文档，则不得指定 --config 参数。

有效位置包括：

- 本地文件路径 (*./component-doc-example.yaml*)。
- S3 URIs (s3://*bucket/key*)。
- Image Builder 组件构建版本 ARNs （arn: aws: imagebuilder: us-west--: component/ /2021.12.02/1 ）。*2:123456789012 my-example-component*

> **Note**
>
> 列表中的项目之间没有空格，只有逗号。

--execution-id

简写形式：-i

这是适用于执行当前 run 命令的唯一 ID。此 ID 包含在输出和日志文件名中，用于唯一标识这些文件，并将它们链接到当前的命令执行。如果省略此设置，则 AWSTOE 生成 GUID。

--log-directory

简写形式：-l

AWSTOE 存储此命令执行的所有日志文件的目标目录。默认情况下，该目录位于以下父目录中：`TOE_<DATETIME>_<EXECUTIONID>`。如果未指定日志目录，则 AWSTOE 使用当前工作目录 (`.`)。

--log-s3-bucket-name

简写形式：-b

如果组件日志存储在 Amazon S3 中（推荐），则将组件应用程序日志 AWSTOE 上传到此参数中命名的 S3 存储桶。

--log-s3-bucket-owner

简写形式：不适用

如果组件日志存储在 Amazon S3 中（推荐），则这是 AWSTOE 写入日志文件的存储桶的所有者账户 ID。

--log-s3-key-prefix

简写形式：-k

如果组件日志存储在 Amazon S3 中（推荐），则这是存储桶中日志位置的 S3 对象键前缀。

--参数 ***name1*** =***value1*** , ***name2***=***value2***...

简写形式：不适用

参数是在组建文档中定义的可变变量，其设置由调用应用程序在运行时提供。

--phases

简写形式：-p

以逗号分隔的列表，它指定要从 YAML 组件文档中运行哪些阶段。如果组件文档包含其他阶段，则这些阶段将无法运行。

--state-directory

简写形式：-s

存储状态跟踪文件的文件路径。

--version

简写形式：-v

指定组件应用程序版本。

选项

--help

简写形式：-h

显示使用组件管理应用程序选项的帮助手册。

--trace

简写形式：-t

启用对控制台的详细日志记录。

awstoe validate command

运行此命令时，它会验证 `--documents` 参数指定的每个组件文档的 YAML 文档语法。

语法

```
awstoe validate [--document-s3-bucket-owner <owner>]
```

```
    --documents <file path,file path,...> [--help] [--trace]
```

参数和选项

参数

--document-s3-bucket-owner

简写形式：不适用

提供的基于 S3 URI 的文档源账户 ID。

--文档 *./doc-1.yaml,./doc-n.yaml*

简写形式：-d *./doc-1.yaml*,*./doc-n*

组件文档（必填）。此参数包含以逗号分隔的文件位置列表，以供运行 YAML 组件文档。有效位置包括：

- 本地文件路径 (*./component-doc-example.yaml*)
- S3 URIs (s3://*bucket/key*)
- Image Builder 组件构建版本 ARNs （arn: aws: imagebuilder: us-west--: component/ /2021.12.02/1 ）*2:123456789012my-example-component*

> **Note**
>
> 列表中的项目之间没有空格，只有逗号。

选项

--help

简写形式：-h

显示使用组件管理应用程序选项的帮助手册。

--trace

简写形式：-t

启用对控制台的详细日志记录。

手动设置以开发自定义组件 AWSTOE

AWS Task Orchestrator and Executor (AWSTOE) 应用程序是一个独立的应用程序，用于在组件定义框架内创建、验证和运行命令。 AWS 服务可用于 AWSTOE 协调工作流程、安装软件、修改系统配置和测试映像构建。

按照以下步骤手动安装 AWSTOE 应用程序，并将其用作独立应用程序来开发自定义组件。如果您使用 Image Builder 控制台或 AWS CLI 命令创建自定义组件，Image Builder 会为您完成这些步骤。有关更多信息，请参阅 使用 Image Builder 创建自定义组件。

验证 AWSTOE 安装下载的签名

本节介绍在基于 Linux、macOS 和 Windows 的操作系统 AWSTOE 上验证安装下载的有效性的推荐流程。

主题

在 Linux 或 macOS 上验证 AWSTOE 安装下载的签名

本主题介绍验证基于 Linux 或 macOS 操作系统的安装下载有效性的推荐流程。 AWSTOE

无论何时从 Internet 下载应用程序，我们都建议您验证软件发布者的身份。另外，请检查应用程序自发布以来是否未被更改或损坏。这会保护您免于安装含有病毒或其他恶意代码的应用程序版本。

如果您在执行本主题中的步骤后确定适用于 AWSTOE 应用程序的软件已遭更改或损坏，请不要运行安装文件。相反，请联系 支持 。有关您的支持选项的更多信息，请参阅支持。

AWSTOE 基于 Linux 和 macOS 操作系统的文件是使用GnuPG安全数字签名的 Pretty Good Privacy (OpenPGP) 标准的开源实现进行签名的。 GnuPG（也称为GPG）通过数字签名提供身份验证和完整性检查。Amazon EC2 发布了您可用于验证下载的 Amazon EC2 CLI 工具的公钥和签名。有关 PGP 和 GnuPG (GPG) 的更多信息，请参阅 http://www.gnupg.org。

第一步是与软件发行商建立信任。下载软件发布者的公有密钥，检查公有密钥的所有人是否真为其人，然后将该公有密钥添加到您的密钥环。密钥环是已知公有密钥的集合。验证公有密钥的真实性后，您可以使用它来验证应用程序的签名。

主题

- 验证并导入公有密钥
- 验证软件包的签名

安装 GPG 工具

如果您的操作系统是 Linux、Unix 或 macOS，GPG 工具很可能已经安装。要测试系统上是否已安装这些工具，请在命令提示符处键入 gpg。如果已安装 GPG 工具，您会看到 GPG 命令提示。如果没有安装 GPG 工具，您会看到错误消息，告诉您无法找到命令。您可以从存储库安装 GnuPG 包。

要安装 GPG 工具，请选择与您的实例匹配的操作系统。

Debian-based Linux

从终端设备运行以下命令：

```
apt-get install gnupg
```

Red Hat–based Linux

从终端设备运行以下命令：

```
yum install gnupg
```

macOS

从终端设备运行以下命令：

```
brew install gnupg
```

验证并导入公有密钥

该过程的下一步是对 AWSTOE 公钥进行身份验证，并将其作为可信密钥添加到您的密GPG钥环中。

验证和导入 AWSTOE 公钥

1. 通过执行下列操作之一获取公共 GPG 生成密钥的副本：

   - 从以下地址下载密钥：

     https://awstoe-**<region>** .s3。 **<region>**.amazonaws。 com/assets/awstoe.gpg。例如 https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/assets/awstoe.gpg。

- 将以下文本中的密钥复制并粘贴到名为 awstoe.gpg 的文件中。确保包含下列所有项：

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQENBF8UqwsBCACdiRF2bkZYaFSDPFC+LIkWLwFvtUCRwAHtD8KIwTJ6LVn3fHAU
GhuK0ZH9mRrqRT2bq/xJjGsnF9VqTj2AJqndGJdDjz75YCZYM+ocZ+r5HSJaeW9i
S5dykHj7Txti2zHe0G5+W0v7v5bPi2sPHsN7XWQ7+G2AMEPTz8PjxY//I0DvMQns
Sle3l9hz6wCClz1l9LbBzTyHfSm5ucTXvNe88XX5Gmt37OCDM7vfli0Ctv8WFoLN
6jbxuA/sV71yIkPm9IYp3+GvaKeT870+sn8/JOOKE/U4sJV1ppbqmuUzDfhrZUaw
8eW8IN9A1FTIuWiZED/5L83UZuQs1S7s2PjlABEBAAG0GkFXU1RPRSA8YXdzdG9l
QGFtYXpvbi5jb20+iQE5BBMBCAAjBQJfFKsLAhsDBwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQ3r3BVvWuvFJGiwf9EVmrBR77+Qe/DUeXZJYoaFr7If/fVDZl
6V3TC6p0J0Veme7uXleRUTFOjzbh+7e5sDX19HrnPquzCnzfMiqbp4lSoeUuNdOf
FcpuTCQH+M+sIEIgPno4PLl0Uj2uE1o++mxmonBl/Krk+hly8hB2L/9n/vW3L7BN
OMb1Ll9PmgGPbWipcT8KRdz4SUex9TXGYzjlWb3jU3uXetdaQY1M3kVKE1siRsRN
YYDtpcjmwbhjpu4xm19aFqNoAHCDctEsXJA/mkU3erwIRocPyjAZE2dnlkL9ZkFZ
z9DQkcIarbCnybDM5lemBbdhXJ6hezJE/b17VA0t1fY04MoEkn6oJg==
=oyze
-----END PGP PUBLIC KEY BLOCK-----
```

2. 在您保存 awstoe.gpg 的目录中的命令提示符处，使用以下命令将 AWSTOE 公钥导入密钥环。

```
gpg --import awstoe.gpg
```

该命令返回的结果类似于下方内容：

```
gpg: key F5AEBC52: public key "AWSTOE <awstoe@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

请记下该键值，因为下一步需要用到。在上一示例中，键值为 F5AEBC52。

3. 通过运行以下命令，将 key-value 替换为上一步中的值来验证指纹：

```
gpg --fingerprint key-value
```

该命令返回的结果类似于下方内容：

```
pub   2048R/F5AEBC52 2020-07-19
      Key fingerprint = F6DD E01C 869F D639 15E5  5742 DEBD C156 F5AE BC52
```

```
uid       [ unknown] AWSTOE <awstoe@amazon.com>
```

此外，指纹字符串应与上述示例中所示的 F6DD E01C 869F D639 15E5 5742 DEBD C156 F5AE BC52 相同。将返回的密钥指纹与此页上发布的指纹进行比较。它们应该相互匹配。如果它们不匹配，请不要安装 AWSTOE 安装脚本，然后与联系 支持。

验证软件包的签名

在安装 GPG 工具、验证并导入 AWSTOE 公有密钥以及确认公有密钥可信后，便可以验证安装脚本的签名。

验证安装脚本签名

1. 在命令提示符处，运行以下命令以下载应用程序的二进制文件：

   ```
   curl -O https://awstoe-<region>.s3.<region>.amazonaws.com/latest/
   linux/<architecture>/awstoe
   ```

   例如：

   ```
   curl -O https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/
   awstoe
   ```

   支持的 **architecture** 值可以是 amd64、386 和 arm64。
2. 在命令提示符处，运行以下命令以下载相同的 S3 key prefix 路径中对应的应用程序二进制文件的签名文件：

   ```
   curl -O https://awstoe-<region>.s3.<region>.amazonaws.com/latest/
   linux/<architecture>/awstoe.sig
   ```

   例如：

   ```
   curl -O https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/
   awstoe.sig
   ```

   支持的 **architecture** 值可以是 amd64、386 和 arm64。
3. 通过在您保存的目录awstoe.sig和 AWSTOE 安装文件中的命令提示符处运行以下命令来验证签名。这两个文件都必须存在。

```
gpg --verify ./awstoe.sig ~/awstoe
```

输出应与以下内容类似：

```
gpg: Signature made Mon 20 Jul 2020 08:54:55 AM IST using RSA key ID F5AEBC52
gpg: Good signature from "AWSTOE awstoe@amazon.com" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F6DD E01C 869F D639 15E5 5742 DEBD C156 F5AE BC52
```

如果输出包含短语 `Good signature from "AWSTOE <awstoe@amazon.com>"`，则意味着已成功验证签名，您可以继续运行 AWSTOE 安装脚本。

如果输出包含短语 `BAD signature`，则检查是否正确执行了此过程。如果您持续获得此响应，不要运行之前下载的安装文件，并且请联系 支持。

下面是有关您可能看到的警告的详细信息：

- 警告：此密钥未使用可信签名进行认证！没有迹象表明签名属于所有者。理想情况下，您可以亲自前往 AWS 办公室领取钥匙。但最可能的情况是，从网站下载此密钥。在这种情况下，该网站就是一个 AWS 网站。
- gpg: no ultimately trusted keys found. 这意味着您 (或您信任的其他人) 对特定密钥不是“绝对信任”。

有关更多信息，请参阅 http://www.gnupg.org。

在 Windows 上验证 AWSTOE 安装下载的签名

本主题介绍在基于 Windows 的操作系统上验证 AWS Task Orchestrator and Executor 应用程序安装文件有效性的推荐流程。

无论何时从 Internet 下载应用程序，我们都建议您验证软件发布者的身份，并检查应用程序从发行以来是否已遭更改或损坏。这会保护您免于安装含有病毒或其他恶意代码的应用程序版本。

如果您在执行本主题中的步骤后确定适用于 AWSTOE 应用程序的软件已遭更改或损坏，请不要运行安装文件。相反，请联系 支持。

要验证基于 Windows 的操作系统上的已下载 awstoe 二进制文件是否有效，必须确保其 Amazon Services LLC 签署人证书的指纹等于此值：

9D CA 72 17 DA FF B8 2F E4 C4 67 77 36 2F A4 AA C9 08 82 15

> **Note**
>
> 在新二进制文件的推出窗口期间，您的签署人证书可能与新的指纹不匹配。如果您的签署人证书不匹配，请验证指纹值是否为：
> BA 81 25 EE AC 64 2E A9 F3 C5 93 CA 6D 3E B7 93 7D 68 75 74

要验证此值，请执行以下过程：

1. 右键单击下载的 `awstoe.exe`，然后打开 Properties (属性) 窗口。
2. 选择数字签名选项卡。
3. 在签名列表 中，选择 Amazon Services LLC，然后选择详细信息。
4. 选择常规选项卡 (如果尚未选择)，然后选择查看证书。
5. 选择详细信息选项卡，然后选择显示下拉列表中的全部 (如果尚未选择)。
6. 向下滚动直至您看到指纹字段，然后选择指纹。这将在下部窗口中显示整个指纹值。
   - 如果下部窗口中的指纹值等于以下值：

     9D CA 72 17 DA FF B8 2F E4 C4 67 77 36 2F A4 AA C9 08 82 15

     那么你下载的 AWSTOE 二进制文件是真实的，可以安全地安装。
   - 如果下部详细信息窗口中的指纹值不等于前述值，请不要运行 `awstoe.exe`。

开始步骤

- 步骤 1：安装 AWSTOE
- 步骤 2：设置 AWS 凭证
- 第 3 步：在本地开发组件文档
- 步骤 4：验证 AWSTOE 组件
- 步骤 5：运行 AWSTOE 组件

步骤 1：安装 AWSTOE

要在本地开发组件，请下载并安装该 AWSTOE 应用程序。

1. 下载 AWSTOE 应用程序

   要进行安装 AWSTOE，请选择适合您的架构和平台的下载链接。有关应用程序下载链接的完整列表，请参阅 AWSTOE 下载

   > **Important**
   >
   > AWS 正在逐步取消对 TLS 版本 1.0 和 1.1 的支持。要访问 S3 存储桶进行 AWSTOE 下载，您的客户端软件必须使用 TLS 版本 1.2 或更高版本。有关更多信息，请参阅此 AWS 安全博客文章。

2. 验证签名

   验证下载的步骤取决于安装 AWSTOE 应用程序后在其中运行应用程序的服务器平台。要在 Linux 服务器上验证您的下载，请参阅 在 Linux 或 macOS 上验证签名。要在 Windows 服务器上验证您的下载，请参阅 在 Windows 上验证签名。

> **Note**
>
> AWSTOE 直接从其下载位置调用。无需单独执行安装步骤。这也意味着 AWSTOE 可以对本地环境进行更改。
> 为确保在组件开发过程中隔离更改，我们建议您使用 EC2 实例来开发和测试 AWSTOE 组件。

步骤 2：设置 AWS 凭证

AWSTOE 运行任务时需要 AWS 凭证才能连接到其他任务 AWS 服务，例如 Amazon S3 和 Amazon CloudWatch，例如：

- 从用户提供的 Amazon S3 路径下载 AWSTOE 文档。
- 运行 `S3Download` 或 `S3Upload` 操作模块。
- 启用后 CloudWatch，将日志流式传输到。

如果您 AWSTOE 在 EC2 实例上运行，则运行 AWSTOE 使用的权限与附加到该 EC2 实例的 IAM 角色相同。

有关的 IAM 角色的更多信息 EC2，请参阅适用于 A mazon 的 IAM 角色EC2。

以下示例说明如何使用AWS_ACCESS_KEY_ID和AWS_SECRET_ACCESS_KEY环境变量设置 AWS 凭证。

要在 Linux、macOS 或 Unix 上设置这些变量，请使用 export。

```
export AWS_ACCESS_KEY_ID=your_access_key_id
```

```
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

要在 Windows 上使用设置这些变量 PowerShell，请使用$env。

```
$env:AWS_ACCESS_KEY_ID=your_access_key_id
```

```
$env:AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

要在 Windows 上使用命令提示设置这些变量，请使用 set。

```
set AWS_ACCESS_KEY_ID=your_access_key_id
```

```
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

第 3 步：在本地开发组件文档

组件是用纯文本 YAML 文档编写的。有关文档语法的更多信息，请参阅 使用 AWSTOE 组件文档框架创建自定义组件。

以下是 Hello World 组件文档示例，可帮助您快速入门。

Linux

本指南中的某些 Linux 组件示例引用名为 hello-world-linux.yml 的组件文档文件。您可以使用下面的文档来着手这些示例。

```
name: Hello World
description: This is hello world testing document for Linux.
schemaVersion: 1.0
phases:
  - name: build
```

```
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the test phase.'
```

Windows

本指南中的某些 Windows 组件示例引用名为 `hello-world-windows.yml` 的组件文档文件。您可以使用下面的文档来着手这些示例。

```
name: Hello World
description: This is Hello World testing document for Windows.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
```

```
            - Write-Host 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the test phase.'
```

macOS

本指南中的某些 macOS 组件示例引用名为 `hello-world-macos.yml` 的组件文档文件。您可以使用下面的文档来着手这些示例。

```
name: Hello World
description: This is hello world testing document for macOS.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the test phase.'
```

步骤 4：验证 AWSTOE 组件

您可以使用 AWSTOE 应用程序在本地验证 AWSTOE 组件的语法。以下示例显示了无需运行组件即可验证其语法的 AWSTOE 应用程序`validate`命令。

> **Note**
>
> AWSTOE 应用程序只能验证当前操作系统的组件语法。例如，在 Windows 上运行 `awstoe.exe` 时，您无法验证使用 `ExecuteBash` 操作模块的 Linux 文档的语法。

Linux 或 macOS

```
awstoe validate --documents /home/user/hello-world.yml
```

Windows

```
awstoe.exe validate --documents C:\Users\user\Documents\hello-world.yml
```

步骤 5：运行 AWSTOE 组件

AWSTOE 应用程序可以使用`--phases`命令行参数运行指定文档的一个或多个阶段。`--phases` 支持的值有 `build`、`validate` 和 `test`。可以用逗号分隔的值形式输入多个阶段值。

当您提供阶段列表时， AWSTOE 应用程序会按顺序运行每个文档的指定阶段。例如， AWSTOE 运行的`build`和`validate`阶段`document1.yaml`，然后运行的`build`和`validate`阶段`document2.yaml`。

为了确保您的日志安全存储并保留以供故障排除，我们建议您在 Amazon S3 中配置日志存储。在 Image Builder 中，用于发布日志的 Amazon S3 位置是在基础设施配置中指定的。有关基础设施配置的更多信息，请参阅 管理 Image Builder 基础设施配置

如果未提供阶段列表，则 AWSTOE 应用程序将按照 YAML 文档中列出的顺序运行所有阶段。

要在单个或多个文档中运行特定阶段，请使用以下命令。

单个阶段

```
awstoe run --documents hello-world.yml --phases build
```

多个阶段

```
awstoe run --documents hello-world.yml --phases build,test
```

文档运行

在单个文档中运行所有阶段

```
awstoe run --documents documentName.yaml
```

在多个文档中运行所有阶段

```
awstoe run --documents documentName1.yaml,documentName2.yaml
```

输入 Amazon S3 信息以从用户定义的本地路径上传 AWSTOE 日志（推荐）

```
awstoe run --documents documentName.yaml --log-s3-bucket-name amzn-s3-demo-destination-
bucket --log-s3-key-prefix S3KeyPrefix --log-s3-bucket-owner S3BucketOwner --log-
directory local_path
```

在单个文档中运行所有阶段，并在控制台上显示所有日志

```
awstoe run --documents documentName.yaml --trace
```

示例命令

```
awstoe run --documents s3://bucket/key/doc.yaml --phases build,validate
```

运行具有唯一 ID 的文档

```
awstoe run --documents documentName.yaml --execution-id user-provided-id --
phases build,test
```

获取以下方面的帮助 AWSTOE

```
awstoe --help
```

使用 AWSTOE 组件文档框架创建自定义组件

要使用 AWS Task Orchestrator and Executor (AWSTOE) 组件框架构建组件，必须提供一个基于 YAML 的文档，该文档表示适用于您创建的组件的阶段和步骤。 AWS 服务 当他们创建新的 Amazon 系统映像 (AMI) 或容器映像时，请使用您的组件。

主题

组件文档工作流程

AWSTOE 组件文档使用阶段和步骤对相关任务进行分组，并将这些任务组织成组件的逻辑工作流程。

> **Tip**
>
> 使用您的组件构建映像的服务可能会实施有关构建过程使用哪些阶段，以及何时允许这些阶段运行的规则。您在设计组件时，请务必考虑这一点。

阶段

阶段表示您的工作流程在映像构建过程中的进展。例如，Image Builder 服务在其构建阶段对其生成的映像使用 `build` 和 `validate` 阶段。它在测试阶段使用 `test` 和 `container-host-test` 阶段来确保在创建最终 AMI 或分发容器映像之前，映像快照或容器映像产生预期的结果。

当组件运行时，每个阶段的关联命令将按照它们在组件文档中出现的顺序进行应用。

阶段规则

- 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。
- 您必须在文档中至少包含以下至少一个阶段：
  - 构建 — 对于 Image Builder，此阶段通常在构建阶段使用。
  - 验证 — 对于 Image Builder，此阶段通常在构建阶段使用。
  - 测试 — 对于 Image Builder，此阶段通常在测试阶段使用。
- 阶段始终按照文档中定义的顺序运行。中为 AWSTOE 命令指定它们的顺序 AWS CLI 无效。

Steps

步骤是定义每个阶段中的工作流程的各个工作单元。步骤按先后顺序运行。但是，一个步骤的输入或输出也可以作为输入馈送到后续步骤中。这就是所谓的“链”。

步骤规则

- 步骤名称对于阶段来说必须是唯一的。
- 步骤必须使用可返回退出代码的支持的操作（操作模块）。

  有关支持的操作模块、其工作原理、 input/output 值和示例的完整列表，请参阅AWSTOE 组件管理器支持的操作模块。

组件日志

AWSTOE 每次运行组件时，都会在 EC2 实例上创建一个新的日志文件夹，用于构建和测试新映像。对于容器映像，日志文件夹存储在容器中。

为了帮助在图像创建过程中出现问题时进行故障排除，在运行组件时 AWSTOE 创建的输入文档和所有输出文件都存储在日志文件夹中。

日志文件夹名称由以下部分组成：

1. 日志目录 — 当服务运行 AWSTOE 组件时，它会与该命令的其他设置一起传入日志目录。在以下示例中，我们展示了 Image Builder 使用的日志文件格式。
   - Linux 和 macOS：`/var/lib/amazon/toe/`
   - Windows：`$env:ProgramFiles\Amazon\TaskOrchestratorAndExecutor\`

2. 文件前缀 — 这是用于所有组件的标准前缀：“TOE_”。
3. 运行时间 — 这是一个 YYYY-MM-DD _HH-MM-SS_UTC-0 格式的时间戳。
4. 执行 ID-这是 AWSTOE 运行一个或多个组件时分配的 GUID。

示例：*/var/lib/amazon/toe/*TOE_*2021-07-01_12-34-56_UTC-0_a1bcd2e3-45f6-789a-bcde-0fa1b2c3def4*

AWSTOE 将以下核心文件存储在日志文件夹中：

输入文件

- document.yaml — 用作命令输入的文档。组件运行后，该文件将作为构件存储。

输出文件

- application.log — 应用程序日志包含来自 AWSTOE 、带有时间戳的调试级别信息，这些信息有关组件运行时发生的情况。
- detailedoutput.json — 此 JSON 文件包含有关组件运行时适用于组件的所有文档、阶段和步骤的运行状态、输入、输出和失败的详细信息。
- console.log — 控制台日志包含组件运行时 AWSTOE 写入控制台的所有标准输出 (stdout) 和标准错误 (stderr) 信息。
- chaining.json — 此 JSON 文件表示 AWSTOE 应用于解析链接表达式的优化。

> **Note**
>
> 日志文件夹还可能包含此处未涵盖的其他临时文件。

输入和输出链

AWSTOE 配置管理应用程序提供了一种通过编写以下格式的参考文献来链接输入和输出的功能：

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

或者

```
{{ 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。

链式规则

- 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。但是，调用服务可能有一些规则，要求链表达式只能在单个阶段的上下文中运行。例如，Image Builder 不支持从构建阶段到测试阶段的链接，因为它独立运行每个阶段。
- 链表达式中的索引从零开始。索引以零 (0) 开头，以引用第一个元素。

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:\sample1.ps1'
          - source: 's3://sample-bucket/sample2.ps1'
            destination: 'C:\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:
            - 'Write-Host "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|Ignore"
        maxAttempts: integer
        inputs:
```

文档的架构定义如下所示。

字段	描述	Type	必需
name	文档的名称。	字符串	否
描述	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	Type	必需
name	阶段的名称。	字符串	是
步骤	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	Type	必需	默认 值
name	用户定义的步骤名称。	字符串		
action	与运行步骤的模块相关的关键字。	字符串		
timeoutSeconds	在失败或重试之前，步骤运行的秒数。 此外，支持 -1 值，表示无限超时。不允许 0 和其他负值。	整数	否	7,200 秒（120 分钟）
onFailure	指定该步骤在失败时应执行的操作。有效值如下所示： • 中止 — 在达到最大尝试次数后该步骤失败，并停	字符串	否	中止

字段	描述	Type	必需	默认 值
	止运行。将阶段和文档的状态设置为 Failed。 • 继续 — 在达到最大尝试次数后该步骤失败，并继续运行剩余的步骤。将阶段和文档的状态设置为 Failed。 • 忽略 — 在达到最大失败尝试次数后将步骤设置为 IgnoredFailure ，并继续运行剩余的步骤。将阶段和文档的状态设置为 SuccessWithIgnoredFailure 。			
maxAttempts	在步骤失败之前允许的最大尝试次数。	整数	否	1

字段	描述	Type	必需	默认 值
inputs	包含操作模块运行步骤所需的参数。	字典	是	

文档示例

以下示例显示了为目标操作系统执行任务的 AWSTOE 组件文档。

Linux

示例 1：运行自定义二进制文件

以下是在 Linux 实例上下载和运行自定义二进制文件的示例文档。

```
name: LinuxBin
description: Download and run a custom Linux binary file.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://<replaceable>amzn-s3-demo-source-bucket</replaceable>/
<replaceable>myapplication</replaceable>
            destination: /tmp/<replaceable>myapplication</replaceable>
      - 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: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

Windows

示例 1：安装 Windows 更新

以下是一个示例文档，用于安装所有可用的 Windows 更新，运行配置脚本，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 run 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:\config.ps1'

      - name: RunConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{build.DownloadConfigScript.inputs[0].destination}}'

      - name: Cleanup
        action: DeleteFile
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - path: '{{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:\testConfig.ps1'

      - name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{validate.DownloadTestConfigScript.inputs[0].destination}}'

      - name: Cleanup
        action: DeleteFile
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - path: '{{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:\testConfig.ps1'
```

```
      - name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{test.DownloadTestConfigScript.inputs[0].destination}}'
```

示例 2：在 Windows 实例 AWS CLI 上安装

以下是使用安装文件在 Windows 实例 AWS CLI 上安装的示例文档。

```
name: InstallCLISetUp
description: Install &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: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

示例 3：使用 MSI 安装程序安装 AWS CLI

以下是使用 MSI 安装程序安装 AWS CLI 的示例文档。

```
name: InstallCLIMSI
description: Install &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: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

macOS

示例 1：运行自定义 macOS 二进制文件

以下是在 macOS 实例上下载和运行自定义二进制文件的示例文档。

```
name: macOSBin
description: Download and run a binary file on macOS.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://<replaceable>amzn-s3-demo-source-bucket</replaceable>/
<replaceable>myapplication</replaceable>
            destination: /tmp/<replaceable>myapplication</replaceable>
      - 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: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

在自定义组件文档中使用变量

变量提供了一种以在整个应用程序中可使用的有意义的名称来标记数据的方法。您可以为复杂的工作流程定义格式简单易读的自定义变量，并在组件的 YAML 应用程序组件文档中引用它们。 AWSTOE

本节提供的信息可帮助您在 YAML 应用程序 AWSTOE 组件文档中为组件定义变量，包括语法、名称约束和示例。

常量

常量是不可变的变量，一旦定义就无法修改或覆盖。常量可以使用 AWSTOE 文档`constants`部分中的值来定义。

常量名称规则

- 名称长度必须介于 3 到 128 个字符之间。
- 该名称只能包含字母/数字字符（a-z、A-Z、0-9）、短划线 (-) 或下划线 (_)。
- 在文档内，此名称必须是唯一的。
- 必须将名称指定为 YAML 字符串。

语法

```
constants:
  - <name>:
```

```
      type: <constant type>
      value: <constant value>
```

键名称	必需	描述
name	是	常量的名称。文档必须是唯一的（不得与任何其他参数名称或常量相同）。
value	是	常量的值。
type	是	常量的类型。支持的类型为 string。

在文档中引用常量值

您可以在 YAML 文档内的步骤或循环输入中引用常量，如下所示：

- 常量引用区分大小写，并且名称必须完全匹配。
- 名称必须用双大括号括{{*MyConstant*}}起来。
- 允许在花括号内留出空格，并且会自动修剪空格。例如，以下所有引用均有效：

  {{ *MyConstant* }}, {{ *MyConstant*}}, {{*MyConstant* }}, {{*MyConstant*}}
- YAML 文档中的引用必须指定为字符串（用单引号或双引号括起来）。

  例如：- {{ *MyConstant* }} 无效，因为它未被标识为字符串。

  但是，以下引用均有效：- '{{ *MyConstant* }}' 和 - "{{ *MyConstant* }}"。

示例

步骤输入中引用的常量

```
name: Download AWS CLI version 2
schemaVersion: 1.0
constants:
  - Source:
      type: string
      value: https://awscli.amazonaws.com/AWSCLIV2.msi
```

```
phases:
  - name: build
    steps:
      - name: Download
        action: WebDownload
        inputs:
          - source: '{{ Source }}'
            destination: 'C:\Windows\Temp\AWSCLIV2.msi'
```

循环输入中引用的常量

```
name: PingHosts
schemaVersion: 1.0
constants:
  - Hosts:
      type: string
      value: 127.0.0.1,amazon.com
phases:
  - name: build
    steps:
      - name: Ping
        action: ExecuteBash
        loop:
          forEach:
            list: '{{ Hosts }}'
            delimiter: ','
        inputs:
          commands:
            - ping -c 4 {{ loop.value }}
```

参数

参数是可变变量，其设置由调用应用程序在运行时提供。您可以在 YAML 文档的 Parameters 部分中定义参数。

参数名称规则

- 名称长度必须介于 3 到 128 个字符之间。
- 该名称只能包含字母/数字字符（a-z、A-Z、0-9）、短划线 (-) 或下划线 (_)。
- 在文档内，此名称必须是唯一的。
- 必须将名称指定为 YAML 字符串。

语法

```
parameters:
  - <name>:
      type: <parameter type>
      default: <parameter value>
      description: <parameter description>
```

键名称	必需	描述
name	是	参数的名称。文档必须是唯一的（不得与任何其他参数名称或常量相同）。
type	是	参数的数据类型。支持的类型包括：string。
default	否	参数的默认值。
description	否	描述参数。

在文档中引用参数值

您可以在 YAML 文档里的步骤或循环输入中引用参数，如下所示：

- 参数引用区分大小写，并且名称必须完全匹配。
- 名称必须用双大括号括{{*MyParameter*}}起来。
- 允许在花括号内留出空格，并且会自动修剪空格。例如，以下所有引用均有效：

  {{ *MyParameter* }}, {{ *MyParameter*}}, {{*MyParameter* }}, {{*MyParameter*}}
- YAML 文档中的引用必须指定为字符串（用单引号或双引号括起来）。

  例如：- {{ *MyParameter* }} 无效，因为它未被标识为字符串。

  但是，以下引用均有效：- '{{ *MyParameter* }}' 和 - "{{ *MyParameter* }}"。

示例

以下示例显示如何在 YAML 文档中使用参数：

- 参考步骤输入中的参数：

```
name: Download AWS CLI version 2
schemaVersion: 1.0
parameters:
  - Source:
      type: string
      default: 'https://awscli.amazonaws.com/AWSCLIV2.msi'
      description: The AWS CLI installer source URL.
phases:
  - name: build
    steps:
      - name: Download
        action: WebDownload
        inputs:
          - source: '{{ Source }}'
            destination: 'C:\Windows\Temp\AWSCLIV2.msi'
```

- 参考循环输入中的参数：

```
name: PingHosts
schemaVersion: 1.0
parameters:
  - Hosts:
      type: string
      default: 127.0.0.1,amazon.com
      description: A comma separated list of hosts to ping.
phases:
  - name: build
    steps:
      - name: Ping
        action: ExecuteBash
        loop:
          forEach:
            list: '{{ Hosts }}'
            delimiter: ','
        inputs:
          commands:
            - ping -c 4 {{ loop.value }}
```

在运行时覆盖参数

您可以将中的--parameters选项与键值对 AWS CLI 一起使用，在运行时设置参数值。

- 将参数键值对指定为名称和值，用等号 (=) 分隔 (<name>=<value>)。
- 多个参数必须用逗号分隔。
- 在 YAML 组件文档中找不到的参数名称将被忽略。
- 参数名称和值都是必需的。

> **Important**
>
> 组件参数是纯文本值，并且已记录在 AWS CloudTrail中。我们建议您使用 AWS Secrets Manager 或 P AWS Systems Manager arameter Store 来存储您的密钥。有关 Secrets Manager 的更多信息，请参阅AWS Secrets Manager 用户指南中的什么是 Secrets Manager?。有关 AWS Systems Manager Parameter Store 的更多信息，请参阅 AWS Systems Manager 用户指南中的 AWS Systems Manager Parameter Store。

语法

```
--parameters name1=value1,name2=value2...
```

CLI 选项	必需	说明
--参数 *name* =*value* ，...	否	此选项采用键/值对列表，以参数名称为键。

示例

以下示例显示如何在 YAML 文档中使用参数：

- 此 --parameter 选项中指定的参数键值对无效：

```
--parameters ntp-server=
```

- 使用 AWS CLI中的 --parameter 选项设置一个参数键值对：

```
--parameters ntp-server=ntp-server-windows-qe.us-east1.amazon.com
```

- 使用 AWS CLI中的 `--parameter` 选项设置多个参数键值对：

```
--parameters ntp-server=ntp-server.amazon.com,http-url=https://internal-us-
east1.amazon.com
```

使用 Systems Manager 参数存储参数

您可以通过在变量前面加上前缀来引用组件文档中的 AWS Systems Manager 参数存储参数（SSM 参数）。`aws:ssm`例如，

`{{ aws:ssm:/my/param }}`解析为 SSM 参数的值。`/my/param`

此功能支持以下 SSM 参数类型：

- 字符串-映射到 AWSTOE 字符串类型。
- StringList — 映射到 AWSTOE `stringList`类型。
- SecureString — 映射到 AWSTOE 字符串类型。

有关参数存储的更多信息，请参阅AWS Systems Manager 用户指南中的AWS Systems Manager 参数存储。

您也可以使用 SSM 参数`SecureString`引用 AWS Secrets Manager 密钥。例如：`{{ aws:ssm:/aws/reference/secretsmanager/test/test-secret }}`。有关更多信息，请参阅从参数存储参数中引用 AWS Secrets Manager 密钥。

> **Important**
>
> Image Builder 从其日志中排除`SecureString`参数解析。但是，您也有责任确保敏感信息不会通过组件文档中发出的命令进行记录。例如，如果您使用带有安全字符串的`echo`命令，则该命令会将纯文本值写入日志。

所需的 IAM 权限

要在组件中使用 Systems Manager 参数，您的实例角色必须拥有参数资源 ARN 的`ssm:GetParameter`权限。例如：

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "ssm:GetParameter",
   "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
  }
 ]
}
```

要访问加密值，您还需要以下权限：

- kms:Decrypt对于使用客户管理的SecureString参数或 AWS Secrets Manager 值进行加密，请添加 AWS KMS key。
- secretsmanager:GetSecretValue如果您引用 Secrets Manager 密钥，请添加。

在组件文档中引用 SSM 参数

以下示例说明如何引用组件中 Systems Manager 参数的 Systems Manager 参数存储参数：

```
name: UseSSMParameterVariable
description: This is a sample component document that prints out the value of an SSM
 Parameter. Never do this for a SecureString parameter.
schemaVersion: 1.0

phases:
  - name: verify
    steps:
      - name: EchoParameterValue
        action: ExecuteBash
        inputs:
          commands:
            - echo "Log SSM parameter name: /my/test/param, value {{ aws:ssm:/my/test/
param }}."
```

SSM 参数的动态运行时变量分辨率

AWSTOE 提供了以下内置函数，您可以在变量引用中使用该函数在运行时操作或转换值。

解析函数

该resolve函数解析另一个变量引用内部的变量引用，从而允许动态变量名引用。这在使用 SSM 参数时很有用，因为其中部分参数路径可能是可变的，并作为文档参数传入。

该resolve函数仅支持 SSM 参数名称部分的动态解析。

语法

以下示例dynamic_variable中的表示 SSM 参数的名称，并且必须是以下参数之一：

- SSM 参数参考（例如，aws:ssm:*/my/param*）
- 组件文档参数引用（例如，*parameter-name*）

```
{{ aws:ssm:resolve(dynamic_variable) }}
```

示例：在运行时解析 SSM 参数

以下示例说明如何在 YAML 组件文档中使用该resolve函数：

```
name: SsmParameterTest
description: This component verifies an SSM parameter variable reference with the echo
 command.
schemaVersion: 1.0

parameters:
  - parameter-name:
      type: string
      description: "test"

phases:
  - name: validate
    steps:
      - name: PrintDynamicVariable
        action: ExecuteBash
        inputs:
          commands:
            - echo "{{ aws:ssm:resolve(parameter-name) }}"
```

在中使用条件构造 AWSTOE

根据指定的条件表达式的计算结果是 `true` 还是 `false`，条件构造会在组件文档中执行不同的操作。您可以使用 `if` 构造来控制组件文档中的执行流程。

if 构造

您可以使用 `if` 构造来评估是否应该运行某步骤。默认情况下，当 `if` 条件表达式的计算结果为 `true` 时， AWSTOE 运行该步骤，当条件表达式计算结果为 `false` 时， AWSTOE 跳过该步骤。如果跳过某个步骤，则在 AWSTOE 评估阶段和文档是否成功运行时，该步骤将被视为成功步骤。

> **Note**
>
> 即使该步骤触发了重启，也只对一个 `if` 语句进行一次计算。如果某个步骤重启，它就会识别出 `if` 语句已经进行计算，并从中断的地方继续进行。

语法

```
if:
  - <conditional expression>:
      [then: <step action>]
      [else: <step action>]
```

键名称	必需	说明
条件表达式	是	条件表达式可以在顶层包含以下类型的运算符之一。 • 比较运算符-有关比较运算符的列表以及它们在 AWSTOE 组件文档中的工作方式的信息，请参阅比较运算符。 • 逻辑运算符 - 逻辑运算符包括 `and`、`or`、和 `not`，并对一个或多个比较运算符进行运算。有关逻辑运算符在

键名称	必需	说明
		AWSTOE 组件文档中的工作原理的更多信息，请参阅逻辑运算符。 如果您的表达式必须满足多个条件，请使用逻辑运算符来指定您的条件。
then	否	定义条件表达式的计算结果为 `true` 时要采取的操作。
else	否	定义条件表达式的计算结果为 `false` 时要采取的操作。
步骤操作	有条件	使用 `then` 或 `else` 时，必须指定以下步骤操作之一： • Abort - AWSTOE 将步骤标记为失败。 • Execute - AWSTOE 运行步骤。 • Skip - AWSTOE 跳过步骤。

示例 1：安装软件包

AWSTOE 组件文档中的以下示例步骤使用逻辑运算符来测试参数值，并在软件包解压缩后运行相应的软件包管理器命令来安装应用程序。

```
    - name: InstallUnzipAptGet
```

```
      action: ExecuteBash
      if:
        and:
            - binaryExists: 'apt-get'
            - not:
                binaryExists: 'unzip'
      inputs:
        commands:
            - sudo apt-get update
            - sudo apt-get install -y unzip

    - name: InstallUnzipYum
      action: ExecuteBash
      if:
        and:
            - binaryExists: 'yum'
            - not:
                binaryExists: 'unzip'
      inputs:
        commands:
            - sudo yum install -y unzip

    - name: InstallUnzipZypper
      action: ExecuteBash
      if:
        and:
            - binaryExists: 'zypper'
            - not:
                binaryExists: 'unzip'
      inputs:
        commands:
            - sudo zypper refresh
            - sudo zypper install -y unzip
```

示例 2：跳过步骤

以下示例显示了跳过步骤的两种方法。一种是使用逻辑运算符，另一种是结合使用比较运算符与 Skip 步骤操作。

```
# Creates a file if it does not exist using not
- name: CreateMyConfigFile-1
  action: ExecuteBash
  if:
```

```
    not:
      fileExists: '/etc/my_config'
  inputs:
    commands:
      - echo "Hello world" > '/etc/my_config'

# Creates a file if it does not exist using then and else
- name: CreateMyConfigFile-2
  action: ExecuteBash
  if:
    fileExists: '/etc/my_config'
    then: Skip
    else: Execute
  inputs:
    commands:
      - echo "Hello world" > '/etc/my_config'
```

在 AWSTOE 组件文档中使用比较运算符

您可以结合使用以下比较运算符与 Assert 操作模块以及使用 if 构造的条件表达式。比较运算符可以对单个值进行比较，例如 `stringIsEmpty`，也可以将基准值与第二个值（变量值）进行比较，以确定条件表达式的计算结果是 `true` 还是 `false`。

如果对两个值进行比较，则第二个值可以是链式变量。

比较不同类型的值时，在比较之前可能会进行以下值转换：

- 对于数值比较，如果变量值是字符串，则在计算之前将字符串 AWSTOE 转换为数字。如果无法进行转换，则比较返回 `false`。例如，如果变量值为 `"1.0"`，则可以进行转换，但如果变量值为 `"a10"`，则转换将会失败。
- 对于字符串比较，如果变量值是一个数字，则在求值之前将其 AWSTOE 转换为字符串。

比较字符串

以下比较运算符结合字符串来比较值、测试空格或空字符串，或者将输入值与正则表达式模式进行比较。字符串比较不区分大小写，也不会从字符串输入的开头或结尾去掉空格。

字符串比较运算符

- stringIsEmpty
- stringIsWhitespace

- stringEquals
- stringLessThan
- stringLessThanEquals
- stringGreaterThan
- stringGreaterThanEquals
- patternMatches

stringIsEmpty

如果指定的字符串不包含任何字符，则 `stringIsEmpty` 运算符将返回 `true`。例如：

```
# Evaluates to true
stringIsEmpty: ""

# Evaluates to false
stringIsEmpty: " "

# Evaluates to false
stringIsEmpty: "Hello."
```

stringIsWhitespace

测试为 `stringIsWhitespace` 指定的字符串是否仅包含空格。例如：

```
# Evaluates to true
stringIsWhitespace: "   "

# Evaluates to false
stringIsWhitespace: ""

# Evaluates to false
stringIsWhitespace: " Hello?"
```

stringEquals

测试为 `stringEquals` 指定的字符串是否与 `value` 参数中指定的字符串完全匹配。例如：

```
# Evaluates to true
stringEquals: 'Testing, testing...'
value: 'Testing, testing...'
```

```
# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'Hello again.'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'TESTING, TESTING....'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: '   Testing, testing...'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'Testing, testing...   '
```

stringLessThan

测试为 `stringLessThan` 指定的字符串是否小于 `value` 参数中指定的字符串。例如：

```
# Evaluates to true
# This comparison operator isn't case sensitive
stringlessThan: 'A'
value: 'a'

# Evaluates to true - 'a' is less than 'b'
stringlessThan: 'b'
value: 'a'

# Evaluates to true
# Numeric strings compare as less than alphabetic strings
stringlessThan: 'a'
value: '0'

# Evaluates to false
stringlessThan: '0'
value: 'a'
```

stringLessThan等于

测试为 `stringLessThanEquals` 指定的字符串是否小于或等于 `value` 参数中指定的字符串。例如：

```
# Evaluates to true - 'a' is equal to 'a'
stringLessThanEquals: 'a'
value: 'a'

# Evaluates to true - since the comparison isn't case sensitive, 'a' is equal to 'A'
stringLessThanEquals: 'A'
value: 'a'

# Evaluates to true - 'a' is less than 'b'
stringLessThanEquals: 'b'
value: 'a'

# Evaluates to true - '0' is less than 'a'
stringLessThanEquals: 'a'
value: '0'

# Evaluates to false - 'a' is greater than '0'
stringLessThanEquals: '0'
value: 'a'
```

stringGreaterThan

测试为 `stringGreaterThan` 指定的字符串是否大于 `value` 参数中指定的字符串。例如：

```
# Evaluates to false - since the comparison isn't case sensitive, 'A' is equal to
 'a'
stringGreaterThan: 'a'
value: 'A'

# Evaluates to true - 'b' is greater than 'a'
stringGreaterThan: 'a'
value: 'b'

# Evaluates to true - 'a' is greater than '0'
stringGreaterThan: '0'
value: 'a'

# Evaluates to false - '0' is less than 'a'
stringGreaterThan: 'a'
value: '0'
```

stringGreaterThan等于

测试为 `stringGreaterThanEquals` 指定的字符串是否大于或等于 `value` 参数中指定的字符串。例如：

```
# Evaluates to true - 'a' is equal to 'A'
stringGreaterThanEquals: 'A'
value: 'a'

# Evaluates to true - 'b' is greater than 'a'
stringGreaterThanEquals: 'a'
value: 'b'

# Evaluates to true - 'a' is greater than '0'
stringGreaterThanEquals: '0'
value: 'a'

# Evaluates to false - '0' is less than 'a'
stringGreaterThanEquals: 'a'
value: '0'
```

patternMatches

测试 `value` 参数中指定的字符串是否与为 `patternMatches` 指定的正则表达式模式匹配。比较使用符合语法的 Golang regexp 包。 RE2 有关 RE2 规则的更多信息，请参阅中的 google/re2 存储库。GitHub

以下示例显示了返回 `true` 的模式匹配：

```
patternMatches: '^[a-z]+$'
value: 'ThisIsValue'
```

比较数字

以下比较运算符适用于数字。根据 YAML 规范，为这些运算符提供的值必须是以下类型之一。对数字比较的支持使用 golang big 包比较运算符，例如：func (*Float) Cmp。

- 整数
- Float（基于 float64，支持从 -1.7e+308 到 +1.7e+308 之间的数字）
- 与以下正则表达式模式相匹配的字符串：`^[-+]?([0-9]+[.])?[0-9]+$`

数字比较运算符

- numberEquals
- numberLessThan
- numberLessThanEquals
- numberGreaterThan
- numberGreaterThanEquals

numberEquals

测试为 numberEquals 指定的数字是否等于 value 参数中指定的数字。以下所有示例的比较均返回 true：

```
# Values provided as a positive number
numberEquals: 1
value: 1

# Comparison value provided as a string
numberEquals: '1'
value: 1

# Value provided as a string
numberEquals: 1
value: '1'

# Values provided as floats
numberEquals: 5.0
value: 5.0

# Values provided as a negative number
numberEquals: -1
value: -1
```

numberLessThan

测试为 numberLessThan 指定的数字是否小于 value 参数中指定的数字。例如：

```
# Evaluates to true
numberLessThan: 2
value: 1
```

```
# Evaluates to true
numberLessThan: 2
value: 1.9

# Evaluates to false
numberLessThan: 2
value: '2'
```

numberLessThan等于

测试为 numberLessThanEquals 指定的数字是否小于或等于 value 参数中指定的数字。例如：

```
# Evaluates to true
numberLessThanEquals: 2
value: 1

# Evaluates to true
numberLessThanEquals: 2
value: 1.9

# Evaluates to true
numberLessThanEquals: 2
value: '2'

# Evaluates to false
numberLessThanEquals: 2
value: 2.1
```

numberGreaterThan

测试为 numberGreaterThan 指定的数字是否大于 value 参数中指定的数字。例如：

```
# Evaluates to true
numberGreaterThan: 1
value: 2

# Evaluates to true
numberGreaterThan: 1
value: 1.1

# Evaluates to false
numberGreaterThan: 1
value: '1'
```

numberGreaterThan等于

测试为 numberGreaterThanEquals 指定的数字是否大于或等于 value 参数中指定的数字。例如：

```
# Evaluates to true
numberGreaterThanEquals: 1
value: 2

# Evaluates to true
numberGreaterThanEquals: 1
value: 1.1

# Evaluates to true
numberGreaterThanEquals: 1
value: '1'

# Evaluates to false
numberGreaterThanEquals: 1
value: 0.8
```

检查文件

以下比较运算符检查文件哈希值或检查文件或文件夹是否存在。

文件和文件夹运算符

- binaryExists
- fileExists
- folderExists
- fileMD5Equals
- fileSHA1Equals
- fileSHA256Equals
- fileSHA512Equals

binaryExists

测试应用程序在当前路径中是否可用。例如：

```
binaryExists: 'foo'
```

> **Note**
>
> 在 Linux 和 macOS 系统上，对于名为的应用程序*foo*，其工作原理与以下 bash 命令相同：type ***foo*** >/dev/null 2>&1，其中$? == 0表示比较成功。
> 在 Windows 系统上，对于名为的应用程序*foo*，其工作原理与$LASTEXITCODE = 0表示成功比较& C:\Windows\System32\where.exe /Q ***foo***的 PowerShell 命令相同。

fileExists

测试指定路径上是否存在文件。您可以提供绝对路径或相对路径。如果您指定的位置存在并且是一个文件，则比较的计算结果为 `true`。例如：

```
fileExists: '/path/to/file'
```

> **Note**
>
> 在 Linux 和 macOS 系统上，其工作原理与以下 bash 命令相同：-d ***/path/to/file***，其中 $? == 0 表示比较成功。
> 在 Windows 系统上，这与 PowerShell 命令的工作原理相同Test-Path -Path '***C:\path\to\file***' -PathType 'Leaf'。

folderExists

测试指定路径上是否存在文件夹。您可以提供绝对路径或相对路径。如果您指定的位置存在并且是一个文件夹，则比较的计算结果为 `true`。例如：

```
folderExists: '/path/to/folder'
```

> **Note**
>
> 在 Linux 和 macOS 系统上，其工作原理与以下 bash 命令相同：-d ***/path/to/folder***，其中 $? == 0 表示比较成功。

在 Windows 系统上，这与 PowerShell 命令的工作原理相同Test-Path -Path '*C:\path\to\folder*' -PathType 'Container'。

文件MD5等于

测试文件的 MD5 哈希值是否等于指定值。例如：

```
fileMD5Equals: '<MD5Hash>'
path: '/path/to/file'
```

文件SHA1等于

测试文件的 SHA1 哈希值是否等于指定值。例如：

```
fileSHA1Equals: '<SHA1Hash>'
path: '/path/to/file'
```

文件SHA256等于

测试文件的 SHA256 哈希值是否等于指定值。例如：

```
fileSHA256Equals: '<SHA256Hash>'
path: '/path/to/file'
```

文件SHA512等于

测试文件的 SHA512 哈希值是否等于指定值。例如：

```
fileSHA512Equals: '<SHA512Hash>'
path: '/path/to/file'
```

在 AWSTOE 组件文档中使用逻辑运算符

您可以使用以下逻辑运算符在组件文档中添加或修改条件表达式。 AWSTOE 按条件的指定顺序计算条件表达式。有关组件文档比较运算符的更多信息，请参阅在 AWSTOE 组件文档中使用比较运算符。

和

使用 and 运算符，您可以将两个或多个比较作为单个表达式进行计算。当列表中的所有条件都为 true 时，表达式的计算结果为 `true`。否则，表达式的计算结果为 `false`。

示例：

下面的示例执行了两个比较：字符串和数字。两个比较的结果均为 true，因此表达式的计算结果为 true。

```
and:
  - stringEquals: 'test_string'
    value: 'test_string'
  - numberEquals: 1
    value: 1
```

以下示例还执行了两个比较。第一个比较的结果为 false，此时计算停止，第二个比较被跳过。表达式的计算结果为 `false`。

```
and:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 1
```

或者

使用 `or` 运算符，您可以将两个或多个比较作为单个表达式进行计算。当其中一个指定的比较结果为 true 时，表达式的计算结果为 `true`。如果指定比较的计算结果均不为 `true`，则表达式的计算结果为 `false`。

示例：

下面的示例执行了两个比较：字符串和数字。第一个比较的结果为 true，因此表达式的计算结果为 `true`，第二个比较被跳过。

```
or:
  - stringEquals: 'test_string'
    value: 'test_string'
  - numberEquals: 1
    value: 3
```

以下示例还执行了两个比较。第一个比较的结果为 false，计算继续进行。第二个比较的结果为 true，因此表达式的计算结果为 `true`。

```
or:
```

```
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 1
```

在最后一个示例中，两个比较的结果均为 false，因此表达式的计算结果为 `false`。

```
or:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 3
```

not

使用 `not` 运算符，您可以否定单个比较。如果比较的结果为 false，则表达式的计算结果为 `true`。如果比较的结果为 true，则表达式的计算结果为 `false`。

示例：

以下示例执行了字符串比较。比较的结果为 false，因此表达式的计算结果为 `true`。

```
not:
  - stringEquals: 'test_string'
    value: 'Hello world!'
```

以下示例还执行了字符串比较。比较的结果为 true，因此表达式的计算结果为 `false`。

```
not:
  - stringEquals: 'test_string'
    value: 'test_string'
```

在中使用循环结构 AWSTOE

本部分提供了有助于您在 AWSTOE中创建循环结构的信息。循环结构定义了重复的指令序列。您可以在 AWSTOE中使用以下类型的循环结构：

- `for` 结构 – 在有界的整数序列上迭代。
- `forEach` 结构
  - `forEach` 使用输入列表循环 – 迭代有限的字符串集合。

- forEach 带分隔列表的循环 – 迭代由分隔符连接的有限字符串集合。

> **Note**
>
> 循环结构仅支持字符串数据类型。

循环结构主题

引用迭代变量

要引用当前迭代变量的索引和值，必须在包含循环结构的步骤的输入主体中使用引用表达式 {{ loop.* }}。此表达式不能用于引用另一个步骤的循环结构的迭代变量。

引用表达式由以下成员组成：

- {{ loop.index }} – 当前迭代的序数位置，索引为 0。
- {{ loop.value }} – 与当前迭代变量关联的值。

循环名称

所有循环结构都有一个用于标识的可选名称字段。如果提供了循环名称，则可以使用它来引用步骤输入主体中的迭代变量。要引用命名循环的迭代索引和值，请在步骤的输入主体中使用带 {{ loop.* }} 的 {{ <loop_name>.* }}。此表达式不能用于引用另一个步骤的命名循环结构。

引用表达式由以下成员组成：

- {{ <loop_name>.index }} – 命名循环当前迭代的序数位置，其索引位于 0。
- {{ <loop_name>.value }} – 与命名循环的当前迭代变量关联的值。

解析引用表达式

解 AWSTOE 析引用表达式如下：

- {{ <loop_name>.* }}— 使用以下逻辑 AWSTOE 解析此表达式：
  - 如果当前正在运行的步骤的循环与 <loop_name> 值匹配，则引用表达式将解析为当前正在运行的步骤的循环结构。
  - 如果命名的循环结构出现在当前运行的步骤内，则 <loop_name> 解析为该结构。
- {{ loop.* }}— 使用在当前运行的步骤中定义的循环结构 AWSTOE 解析表达式。

如果在不包含循环的步骤中使用引用表达式，则 AWSTOE 不会解析这些表达式，并且它们出现在步骤中而不进行替换。

> **Note**
>
> 引用表达式必须用双引号括起来，YAML 编译器才能正确解释。

循环结构的类型

本节提供有关可在 AWSTOE中使用的循环结构类型的信息和示例。

循环结构类型

- for 循环
- forEach 使用输入列表循环
- forEach 使用带分隔列表的循环

for 循环

该 for 循环迭代处于由变量开头和结尾勾勒的边界内指定的整数范围。迭代值在集合 [start, end] 中，包括边界值。

AWSTOE 验证startend、和updateBy值，以确保组合不会导致无限循环。

for 循环架构

```
- name: "StepName"
  action: "ActionModule"
  loop:
    name: "string"
    for:
      start: int
      end: int
```

```
        updateBy: int
inputs:
  ...
```

for 循环输入

字段	描述	Type	必需	默认
name	循环的唯一名称。与同一阶段的其他循环名称相比，它必须是唯一的。	字符串	否	""
start	迭代的起始值。不接受链接表达式。	整数	是	不适用
end	迭代的结束值。不接受链接表达式。	整数	是	不适用
updateBy	通过加法更新迭代值的差异。它必须是非零负值或正值。不接受链接表达式。	整数	是	不适用

for 循环输入示例

```
  - name: "CalculateFileUploadLatencies"
    action: "ExecutePowerShell"
    loop:
      for:
        start: 100000
        end: 1000000
        updateBy: 100000
    inputs:
      commands:
        - |
```

```
          $f = new-object System.IO.FileStream c:\temp\test{{ loop.index }}.txt,
 Create, ReadWrite
          $f.SetLength({{ loop.value }}MB)
          $f.Close()
        - c:\users\administrator\downloads\latencyTest.exe --file c:\temp
\test{{ loop.index }}.txt
        - AWS s3 cp c:\users\administrator\downloads\latencyMetrics.json s3://bucket/
latencyMetrics.json
        - |
          Remove-Item -Path c:\temp\test{{ loop.index }}.txt
          Remove-Item -Path c:\users\administrator\downloads\latencyMetrics.json
```

forEach 使用输入列表循环

该 forEach 循环迭代一个显式的值列表，该列表可以是字符串和链式表达式。

forEach 使用输入列表架构进行循环

```
  - name: "StepName"
    action: "ActionModule"
    loop:
      name: "string"
      forEach:
        - "string"
    inputs:
  ...
```

forEach 使用输入列表输入进行循环

字段	描述	Type	必需	默认
name	循环的唯一名称。与同一阶段的其他循环名称相比，它必须是唯一的。	字符串	否	""
forEach 循环字符串列表	用于迭代的字符串列表。接受链式表达式作为列表中的字符串	字符串列表	是	不适用

字段	描述	Type	必需	默认
	。链式表达式必须用双引号括起来，YAML 编译器才能正确解释它们。			

forEach 使用输入列表循环示例 1

```
- name: "ExecuteCustomScripts"
  action: "ExecuteBash"
  loop:
    name: BatchExecLoop
    forEach:
      - /tmp/script1.sh
      - /tmp/script2.sh
      - /tmp/script3.sh
  inputs:
    commands:
      - echo "Count {{ BatchExecLoop.index }}"
      - sh "{{ loop.value }}"
      - |
        retVal=$?
        if [ $retVal -ne 0 ]; then
          echo "Failed"
        else
          echo "Passed"
       fi
```

forEach 使用输入列表循环示例 2

```
- name: "RunMSIWithDifferentArgs"
  action: "ExecuteBinary"
  loop:
    name: MultiArgLoop
    forEach:
      - "ARG1=C:\Users ARG2=1"
      - "ARG1=C:\Users"
      - "ARG1=C:\Users ARG3=C:\Users\Administrator\Documents\f1.txt"
  inputs:
```

```
    commands:
      path: "c:\users\administrator\downloads\runner.exe"
      args:
        - "{{ MultiArgLoop.value }}"
```

forEach 使用输入列表循环示例 3

```
- name: "DownloadAllBinaries"
  action: "S3Download"
  loop:
    name: MultiArgLoop
    forEach:
      - "bin1.exe"
      - "bin10.exe"
      - "bin5.exe"
  inputs:
    - source: "s3://bucket/{{ loop.value }}"
      destination: "c:\temp\{{ loop.value }}"
```

forEach 使用带分隔列表的循环

该循环遍历包含由分隔符分隔的值的字符串。要遍历字符串的组成部分，请 AWSTOE 使用分隔符将字符串拆分为适合迭代的数组。

forEach 使用分隔列表架构的循环

```
- name: "StepName"
  action: "ActionModule"
  loop:
    name: "string"
    forEach:
      list: "string"
      delimiter: ".,;:\n\t -_"
  inputs:
...
```

forEach 使用分隔列表输入的循环

字段	描述	Type	必需	默认
name	赋予循环的唯一名称。与同一阶	字符串	否	""

字段	描述	Type	必需	默认
	段的其他循环名称相比，它应该是唯一的。			
`list`	由组成字符串组成的字符串，这些字符串由一个通用分隔符连接在一起。也接受链式表达式。如果是链式表达式，请确保用双引号将它们括起来，以便 YAML 编译器正确解释。	字符串	是	不适用

字段	描述	Type	必需	默认
delimiter	用于在区块中分隔字符串的字符。默认为逗号字符。在给定的列表中只允许使用一个分隔符： • 圆点："." • 逗号："," • 分号：";" • 冒号：":" • 新建行："\n" • 选项卡："\t" • 空格：" " • 连字符："-" • 下划线："_" 不能使用链接表达式。	字符串	否	逗号：","

> **Note**
>
> list 的值被视为不可变的字符串。如果在运行时更改了 list 的源，则它在运行期间不会反映出来。

forEach 使用分隔列表循环示例 1

此示例使用以下链接表达式模式来引用另一个步骤的输出：<phase_name>.<step_name>.[inputs | outputs].<var_name>。

```
- name: "RunMSIs"
  action: "ExecuteBinary"
  loop:
    forEach:
      list: "{{ build.GetAllMSIPathsForInstallation.outputs.stdout }}"
      delimiter: "\n"
  inputs:
    commands:
      path: "{{ loop.value }}"
```

forEach 使用分隔列表循环示例 2

```
- name: "UploadMetricFiles"
  action: "S3Upload"
  loop:
    forEach:
      list: "/tmp/m1.txt,/tmp/m2.txt,/tmp/m3.txt,..."
  inputs:
    commands:
      - source: "{{ loop.value }}"
        destination: "s3://bucket/key/{{ loop.value }}"
```

步骤字段

循环是步骤的一部分。任何与步骤运行相关的字段都不会应用于单个迭代。步骤字段仅应用于步骤级别，如下所示：

- timeoutSeconds – 循环的所有迭代都必须在此字段指定的时间段内运行。如果循环运行超时，则 AWSTOE 运行该步骤的重试策略，并为每次新的尝试重置超时参数。如果循环运行在达到最大重试次数后超过超时值，则该步骤的失败消息将表明循环运行已超时。
- onFailure – 对步骤应用失败处理，如下所示：
  - 如果 onFa ilure 设置为Abort，则 AWSTOE 退出循环并根据重试策略重试该步骤。在达到最大重试次数后，将当前步骤 AWSTOE 标记为失败，并停止运行该进程。

    AWSTOE 将父阶段和文档的状态码设置为Failed。

> **Note**
>
> 在失败的步骤之后，不再运行任何其他步骤。

- 如果 onFailure 设置为 `Continue`，则 AWSTOE 退出循环并根据重试策略重试该步骤。在达到最大重试次数后，将当前步骤 AWSTOE 标记为失败，然后继续运行下一个步骤。

  AWSTOE 将父阶段和文档的状态码设置为`Failed`。

- 如果 onFailure 设置为 `Ignore`，则 AWSTOE 退出循环并根据重试策略重试该步骤。在达到最大重试次数后，将当前步骤 AWSTOE 标记为`IgnoredFailure`，然后继续运行下一步。

  AWSTOE 将父阶段和文档的状态码设置为`SuccessWithIgnoredFailure`。

> **Note**
>
> 这仍被视为成功运行，但包含一些信息，可让您知道一个或多个步骤失败并被忽略。

- maxAttempts – 每次重试，整个步骤和所有迭代都是从头开始运行的。
- 状态 – 步骤运行的总体状态。`status` 不代表各个迭代的状态。包含循环的步骤状态确定如下：
  - 如果单个迭代运行失败，则步骤的状态指向失败。
  - 如果所有迭代都成功，则步骤的状态指向成功。
- startTime – 步骤运行的总开始时间。不代表各个迭代的开始时间。
- endTime – 步骤运行的总结束时间。不代表单个迭代的结束时间。
- failureMessage – 包括在出现非超时错误时失败的迭代索引。如果出现超时错误，消息指出该循环运行失败。为了最大限度地减少失败消息的大小，不为每次迭代提供单独的错误消息。

步骤和迭代输出

每次迭代都包含一个输出。在循环运行结束时， AWSTOE 将所有成功的迭代输出合并到中。`detailedOutput.json`合并输出是属于相应输出键的值的排序规则，这些值在操作模块的输出架构中定义。下面的示例说明如何合并输出：

迭代 1 的 **`ExecuteBash`** 的输出

```
{
 "stdout":"Hello"
```

```
}
```

迭代 2 的 **ExecuteBash** 的输出

```
{
 "stdout":"World"
}
```

步骤的 **ExecuteBash** 的输出

```
{
 "stdout":"Hello\nWorld"
}
```

例如，ExecuteBash、ExecutePowerShell 和 ExecuteBinary 是操作模块，其返回 STDOUT 作为操作模块输出。STDOUT 消息与换行符连接以在 detailedOutput.json 中生成步骤的总体输出。

AWSTOE 不会合并失败迭代的输出。

AWSTOE 组件管理器支持的操作模块

映像构建服务（例如 EC2 Image Builder）使用 AWSTOE 操作模块来帮助配置用于构建和测试自定义机器映像的 EC2 实例。本节介绍常用 AWSTOE 操作模块的功能以及如何配置它们，包括示例。

组件是用纯文本 YAML 文档编写的。有关文档语法的更多信息，请参阅 使用 AWSTOE 组件文档框架创建自定义组件。

> **Note**
>
> 所有操作模块在运行时都使用与 Systems Manager 代理相同的帐户，即 Linux 上的 root 和 Windows 上的 NT Authority\SYSTEM。

以下交叉引用按操作模块执行的操作类型对其进行了分类。

一般执行

- Assert（Linux、Windows、macOS）

- ExecuteBash（Linux、macOS）
- ExecuteBinary（Linux、Windows、macOS）
- ExecuteDocument（Linux、Windows、macOS）
- ExecutePowerShell（视窗）

文件下载与上传

- S3Download（Linux、Windows、macOS）
- S3Upload（Linux、Windows、macOS）
- WebDownload（Linux、Windows、macOS）

文件系统操作

- AppendFile（Linux、Windows、macOS）
- CopyFile（Linux、Windows、macOS）
- CopyFolder（Linux、Windows、macOS）
- CreateFile（Linux、Windows、macOS）
- CreateFolder（Linux、Windows、macOS）
- CreateSymlink（Linux、Windows、macOS）
- DeleteFile（Linux、Windows、macOS）
- DeleteFolder（Linux、Windows、macOS）
- ListFiles（Linux、Windows、macOS）
- MoveFile（Linux、Windows、macOS）
- MoveFolder（Linux、Windows、macOS）
- ReadFile（Linux、Windows、macOS）
- SetFileEncoding（Linux、Windows、macOS）
- SetFileOwner（Linux、Windows、macOS）
- SetFolderOwner（Linux、Windows、macOS）

- SetFilePermissions（Linux、Windows、macOS）
- SetFolderPermissions（Linux、Windows、macOS）

软件安装操作

- InstallMSI (Windows)
- UninstallMSI (Windows)

系统操作

- Reboot（Linux、Windows）
- SetRegistry（视窗）
- UpdateOS（Linux、Windows）

通用执行模块

下一部分详细介绍了运行命令和控制执行工作流的操作模块。

常规执行操作模块

- Assert（Linux、Windows、macOS）
- ExecuteBash（Linux、macOS）
- ExecuteBinary（Linux、Windows、macOS）
- ExecuteDocument（Linux、Windows、macOS）
- ExecutePowerShell（视窗）

Assert（Linux、Windows、macOS）

Assert 操作模块使用比较运算符或逻辑运算符作为输入进行值比较。运算符表达式的结果（true 或 false）表示步骤的总体成功或失败状态。

如果比较或逻辑运算符表达式的计算结果为 `true`，则该步骤将标记为 `Success`。否则，该步骤将标记为 `Failed`。如果步骤失败，则 `onFailure` 参数决定步骤的结果。

Input

键名称	说明	Type	必需
input	包含单个比较运算符或逻辑运算符。注意，逻辑运算符可以包含多个比较运算符。	这不是固定不变的，具体取决于操作人员	是

输入示例：使用 **stringEquals** 比较运算符进行简单比较

此示例的计算结果为 true。

```
- name: StringComparison
  action: Assert
  inputs:
    stringEquals: '2.1.1'
    value: '{{ validate.ApplicationVersion.outputs.stdout }}'
```

输入示例：使用 **patternMatches** 比较运算符的正则表达式比较

这些示例的计算结果均为 true。

```
- name: Letters only
  action: Assert
  inputs:
    patternMatches: '^[a-zA-Z]+$'
    value: 'ThisIsOnlyLetters'

- name: Letters and spaces only
  action: Assert
  inputs:
    patternMatches: '^[a-zA-Z\s]+$'
    value: 'This text contains spaces'

- name: Numbers only
  action: Assert
  inputs:
    patternMatches: '^[0-9]+$'
    value: '1234567890'
```

输入示例：使用逻辑运算符和链式变量的嵌套比较

以下示例演示了使用逻辑运算符和链式变量执行嵌套比较。如果以下任一条件为 true，则 Assert 的计算结果为 true：

- ApplicationVersion 大于 2.0 并且 CPUArchitecture 等于 arm64。
- CPUArchitecture 等于 x86_64。

```
- name: NestedComparisons
  action: Assert
  inputs:
    or: # <- first level deep
      - and: # <- second level deep
          - numberGreaterThan: 2.0 # <- third level deep
            value: '{{ validate.ApplicationVersion.outputs.stdout }}'
          - stringEquals: 'arm64'
            value: '{{ validate.CPUArchitecture.outputs.stdout }}'
      - stringEquals: 'x86_64'
        value: '{{ validate.CPUArchitecture.outputs.stdout }}'
```

输出：

Assert 的输出表示步骤的成功或失败。

ExecuteBash（Linux、macOS）

ExecuteBash操作模块允许您使用内联 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都将转换为文件（例如 input.sh）并使用 bash Shell 执行。Shell 文件的执行结果是退出代码步骤。

如果脚本退出时退出代码为，则该ExecuteBash模块会处理系统重新启动。194在启动后，该应用程序执行以下操作之一：

- 如果是由 Systems Manager 代理执行的，该应用程序将向调用方返回退出代码。Systems Manager 代理处理系统重启并运行启动重启的步骤，如从脚本重启托管实例中所述。
- 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，然后重新启动系统。

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。

Input

键名称	说明	Type	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：重启前后

```
name: ExitCode194Example
description: This shows how the exit code can be used to restart a system with
 ExecuteBash
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: RestartTrigger
        action: ExecuteBash
        inputs:
          commands:
            - |
              REBOOT_INDICATOR=/var/tmp/reboot-indicator
              if [ -f "${REBOOT_INDICATOR}" ]; then
                echo 'The reboot file exists. Deleting it and exiting with success.'
                rm "${REBOOT_INDICATOR}"
                exit 0
              fi
              echo 'The reboot file does not exist. Creating it and triggering a
 restart.'
              touch "${REBOOT_INDICATOR}"
              exit 194
```

Output

字段	描述	Type
stdout	命令执行的标准输出。	字符串

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，构建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，构建过程可能会失败。

输出示例：重启前（首次按照文档）

```
{
 “stdout”: “The reboot file does not exist. Creating it and triggering a restart."
}
```

输出示例：重启后（第二次按照文档）

```
{
 “stdout”: “The reboot file exists. Deleting it and exiting with success."
}
```

ExecuteBinary（Linux、Windows、macOS）

ExecuteBinary操作模块允许您使用命令行参数列表运行二进制文件。

如果二进制文件退出时退出代码为 194 (Linux) 或 3010 (Windows)，则该ExecuteBinary模块会处理系统的重新启动。在触发后，该应用程序执行以下操作之一：

- 如果是由 Systems Manager 代理执行的，该应用程序将向调用方返回退出代码。Systems Manager 代理负责系统重启并运行启动重启的步骤，如从脚本重启托管实例中所述。
- 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，然后重新启动系统。

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。

Input

键名称	说明	Type	必需
path	要执行的二进制文件的路径。	字符串	是
arguments	包含在运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：安装 .NET

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

Output

字段	描述	Type
stdout	命令执行的标准输出。	字符串

输出示例

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

ExecuteDocument（Linux、Windows、macOS）

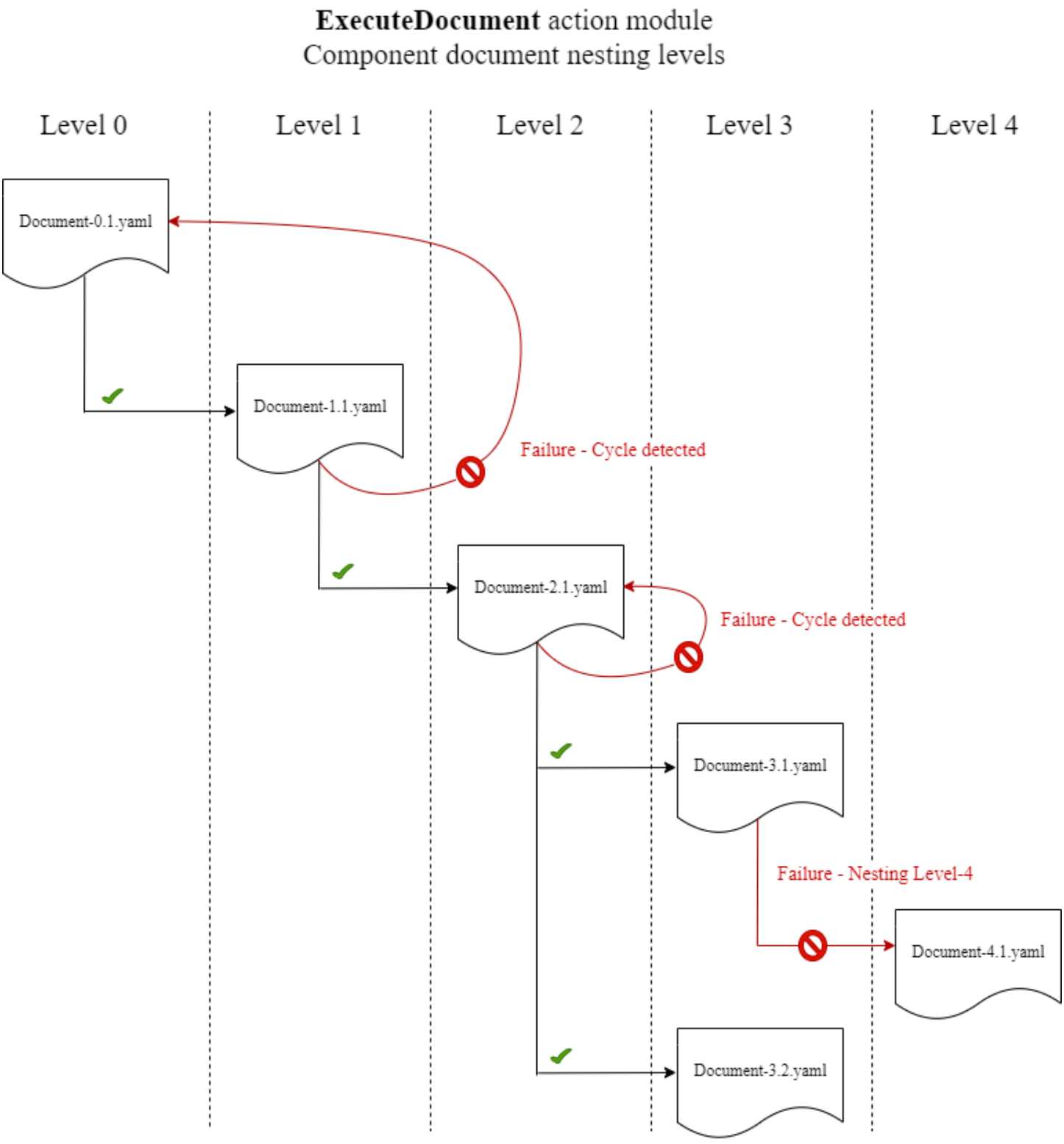
ExecuteDocument操作模块增加了对嵌套组件文档的支持，从一个文档中运行多个组件文档。AWSTOE 验证运行时在输入参数中传递的文档。

限制

- 此操作模块运行一次，不允许重试，也没有设置超时限制的选项。 ExecuteDocument设置以下默认值，如果您尝试更改这些值，则会返回错误。
  - `timeoutSeconds`: -1
  - `maxAttempts`：1

  > **Note**
  >
  > 您可以将这些值留空，并 AWSTOE 使用默认值。

- 文档可以嵌套，最多可嵌套三层，不得超过此限制。三个嵌套级别转换为四个文档级别，因为顶层不是嵌套的。在这种情况下，最低级别的文档不得调用任何其他文档。
- 不允许循环执行组件文档。任何在循环结构之外调用自身文档，或者调用当前执行链中调用更高层的文档，都会引发循环，从而引发无限循环。当检测到循环执行时， AWSTOE 会停止执行并记录失败。



如果组件文档尝试自行运行，或者尝试运行当前执行链中更高层的任何组件文档，执行将会失败。

输入

键名称	说明	Type	必需
document	组件文档的路径。有效选项包括： • 本地文件路径 • S3 URIs • EC2 Image Builder 组件构建版本 ARNs	字符串	是
document-s3-bucket-owner	S3 存储桶所有者的账户 ID，用于存储组件文档的 S3 存储桶。（如果您在组件文档 URIs 中使用 S3，则建议使用。）	字符串	否
phases	组件文件中要运行的阶段已用逗号分隔的列表来表示。如果未指定任何阶段，将运行所有阶段。	字符串	否
parameters	在运行时作为键值对传递到组件文档的输入参数。	参数映射列表	否

参数映射输入

键名称	说明	Type	必需
name		字符串	是

键名称	说明	Type	必需
	要传递给ExecuteDocument操作模块正在运行的组件文档的输入参数的名称。		
value	输入参数的值。	字符串	是

输入示例

以下示例显示了组件文档的输入变体，具体取决于您的安装路径。

输入示例：本地文档路径

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: Sample-1.yaml
          phases: build
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

输入示例：S3 URI 为文档路径

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
```

```
        action: ExecuteDocument
        inputs:
          document: s3://my-bucket/Sample-1.yaml
          document-s3-bucket-owner: 123456789012
          phases: build,validate
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

输入示例：EC2 Image Builder 组件 ARN 为文档路径

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: arn:aws:imagebuilder:us-west-2:aws:component/Sample-Test/1.0.0
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

使用 ForEach 循环运行文档

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        loop:
          name: 'myForEachLoop'
          forEach:
```

```
            - Sample-1.yaml
            - Sample-2.yaml
        inputs:
          document: "{{myForEachLoop.value}}"
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

使用 For 循环运行文档

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        loop:
          name: 'myForLoop'
          for:
            start: 1
            end: 2
            updateBy: 1
        inputs:
          document: "Sample-{{myForLoop.value}}.yaml"
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

Output

AWSTOE 创建`detailedoutput.json`每次运行时都调用的输出文件。该文件包含运行时调用的每个组件文档的每个阶段和步骤的详细信息。对于ExecuteDocument操作模块，您可以在`outputs`字段中找到简短的运行时摘要，以及有关该模块在其中运行的阶段、步骤和文档的详细信息`detailedOutput`。

```
{
 \"executedStepCount\":1,\"executionId\":\"97054e22-06cc-11ec-9b14-acde48001122\",
\"failedStepCount\":0,\"failureMessage\":\"\",\"ignoredFailedStepCount\":0,\"logUrl\":
\"\",\"status\":\"success\"
}",
```

每个组件文档的输出摘要对象都包含以下详细信息以及示例值，如下所示：

- executedStepCount“:1
- "executionId": "12345a67-89bc-01de-2f34-abcd56789012"
- failedStepCount“: 0
- "failureMessage": ""
- “ignoredFailedStep计数”: 0
- "logUrl": ""
- "status": "success"

输出示例

以下示例显示了发生嵌套执行时ExecuteDocument操作模块的输出。在此示例中，main.yaml 组件文档成功运行了 Sample-1.yaml 组件文档。

```
{
    "executionId": "12345a67-89bc-01de-2f34-abcd56789012",
    "status": "success",
    "startTime": "2021-08-26T17:20:31-07:00",
    "endTime": "2021-08-26T17:20:31-07:00",
    "failureMessage": "",
    "documents": [
        {
            "name": "",
            "filePath": "main.yaml",
            "status": "success",
            "description": "",
            "startTime": "2021-08-26T17:20:31-07:00",
            "endTime": "2021-08-26T17:20:31-07:00",
            "failureMessage": "",
            "phases": [
                {
                    "name": "build",
```

```
                    "status": "success",
                    "startTime": "2021-08-26T17:20:31-07:00",
                    "endTime": "2021-08-26T17:20:31-07:00",
                    "failureMessage": "",
                    "steps": [
                        {
                            "name": "ExecuteNestedDocument",
                            "status": "success",
                            "failureMessage": "",
                            "timeoutSeconds": -1,
                            "onFailure": "Abort",
                            "maxAttempts": 1,
                            "action": "ExecuteDocument",
                            "startTime": "2021-08-26T17:20:31-07:00",
                            "endTime": "2021-08-26T17:20:31-07:00",
                            "inputs": "[{\"document\":\"Sample-1.yaml\",\"document-s3-
bucket-owner\":\"\",\"phases\":\"\",\"parameters\":null}]",
                            "outputs": "[{\"executedStepCount\":1,\"executionId\":
\"98765f43-21ed-09cb-8a76-fedc54321098\",\"failedStepCount\":0,\"failureMessage\":\"\",
\"ignoredFailedStepCount\":0,\"logUrl\":\"\",\"status\":\"success\"}]",
                            "loop": null,
                            "detailedOutput": [
                                {
                                    "executionId": "98765f43-21ed-09cb-8a76-
fedc54321098",
                                    "status": "success",
                                    "startTime": "2021-08-26T17:20:31-07:00",
                                    "endTime": "2021-08-26T17:20:31-07:00",
                                    "failureMessage": "",
                                    "documents": [
                                        {
                                            "name": "",
                                            "filePath": "Sample-1.yaml",
                                            "status": "success",
                                            "description": "",
                                            "startTime": "2021-08-26T17:20:31-07:00",
                                            "endTime": "2021-08-26T17:20:31-07:00",
                                            "failureMessage": "",
                                            "phases": [
                                                {
                                                    "name": "build",
                                                    "status": "success",
                                                    "startTime":
 "2021-08-26T17:20:31-07:00",
```

```
                                                    "endTime":
 "2021-08-26T17:20:31-07:00",
                                                    "failureMessage": "",
                                                    "steps": [
                                                        {
                                                            "name": "ExecuteBashStep",
                                                            "status": "success",
                                                            "failureMessage": "",
                                                            "timeoutSeconds": 7200,
                                                            "onFailure": "Abort",
                                                            "maxAttempts": 1,
                                                            "action": "ExecuteBash",
                                                            "startTime":
 "2021-08-26T17:20:31-07:00",
                                                            "endTime":
 "2021-08-26T17:20:31-07:00",
                                                            "inputs": "[{\"commands\":
[\"echo \\\"Hello World!\\\"\"]}]",
                                                            "outputs": "[{\"stdout\":
\"Hello World!\"}]",
                                                            "loop": null,
                                                            "detailedOutput": null
                                                        }]
                                                }]
                                        }]
                                }]
                        }]

                }]
        }]
}
```

ExecutePowerShell（视窗）

ExecutePowerShell操作模块允许您使用内联 shell 代码/命令运行 PowerShell 脚本。该模块支持 Windows 平台和 Windows PowerShell。

命令块中 commands/instructions 指定的所有内容都将转换为脚本文件（例如`input.ps1`），并使用 Windows 运行PowerShell。Shell 文件的执行结果是退出代码。

如果 shell 命令退出时退出代码为，则该ExecutePowerShell模块将处理系统重新启动。`3010`在启动后，该应用程序执行以下操作之一：

- 如果由 Systems Manager 代理运行，则将退出代码交给调用者。Systems Manager 代理处理系统重启并运行启动重启的步骤，如从脚本重启托管实例中所述。
- 保存当前 executionstate，配置重新启动触发器以重新运行该应用程序，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。

Input

键名称	说明	Type	必需
commands	包含要按照PowerShell 语法运行的指令或命令的列表。允许使用多行 YAML。	字符串列表	可以。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命令行参数对这个文件运行。路径必须指向 .ps1 文件。	字符串	可以。必须指定 commands 或 file，但不能同时指定。

输入示例：重启前后

```
name: ExitCode3010Example
description: This shows how the exit code can be used to restart a system with
 ExecutePowerShell
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: RestartTrigger
        action: ExecutePowerShell
        inputs:
          commands:
            - |
              $rebootIndicator = Join-Path -Path $env:SystemDrive -ChildPath 'reboot-
indicator'
```

```
            if (Test-Path -Path $rebootIndicator) {
              Write-Host 'The reboot file exists. Deleting it and exiting with
success.'
              Remove-Item -Path $rebootIndicator -Force | Out-Null
              [System.Environment]::Exit(0)
            }
            Write-Host 'The reboot file does not exist. Creating it and triggering a
restart.'
            New-Item -Path $rebootIndicator -ItemType File | Out-Null
            [System.Environment]::Exit(3010)
```

Output

字段	描述	Type
stdout	命令执行的标准输出。	字符串

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，构建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，构建过程可能会失败。

输出示例：重启前（首次按照文档）

```
{
 “stdout”: “The reboot file does not exist. Creating it and triggering a restart."
}
```

输出示例：重启后（第二次按照文档）

```
{
 “stdout”: “The reboot file exists. Deleting it and exiting with success."
}
```

文件下载与上传模块

下一节详细介绍了上传或下载文件的操作模块。

下载与上传操作模块

- S3Download（Linux、Windows、macOS）
- S3Upload（Linux、Windows、macOS）
- WebDownload （Linux、Windows、macOS）

S3Download（Linux、Windows、macOS）

使用 S3Download 操作模块，您可以将 Amazon S3 对象或一组对象下载到您使用 destination 路径指定的本地文件或文件夹。如果指定位置已存在任何文件，且 overwrite 标志设置为正确，则 S3Download 会覆盖该文件。

source 位置可以指向 Amazon S3 中的特定对象，也可以使用带有星号通配符 (*) 的键前缀，以下载一组与键前缀路径匹配的对象。当您在 source 位置指定键前缀时，S3Download 操作模块会下载与该前缀匹配的所有内容（包括文件和文件夹）。确保键前缀以正斜杠结尾，后跟星号 (/*)，以下载与该前缀匹配的所有内容。例如：*s3://my-bucket/my-folder/**。

如果在下载过程中对指定键前缀的 S3Download 操作失败，文件夹内容不会回滚到失败之前的状态。目标文件夹将保持失败时的状态。

支持的使用案例

S3Download 操作模块支持以下案例：

- Amazon S3 对象将下载到下载路径中指定的本地文件夹。
- Amazon S3 对象（在 Amazon S3 文件路径中带有键前缀）将下载到指定的本地文件夹，该文件夹以递归方式将所有与键前缀匹配的 Amazon S3 对象复制到本地文件夹。

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：s3:GetObject反对 bucket/object（例如，arn:aws:s3:::**BucketName**/*）。
- 多个文件：s3:ListBucket针对 bucket/object（例如，arn:aws:s3:::**BucketName**）和s3:GetObject反对 bucket/object（例如arn:aws:s3:::**BucketName**/*）。

Input

Key	说明	Type	必需	默认
source	Amazon S3 存储桶为下载源。您可以指定特定	字符串	是	不适用

Key	说明	Type	必需	默认
	对象的路径，也可以使用键前缀（以正斜杠结尾，后跟星号通配符 (/*) 来下载一组与键前缀匹配的对象。			
destination	下载 Amazon S3 对象的本地路径。要下载单个文件，您必须将文件名指定为路径的一部分。例如 */myfolder/package.zip* 。	字符串	是	不适用
expectedBucketOwner	source 路径中提供的存储桶的预期拥有者账户 ID。我们建议您验证源中指定的 Amazon S3 存储桶的所有权。	字符串	否	不适用

Key	说明	Type	必需	默认
overwrite	设置为正确时，如果指定本地路径的目标文件夹中已存在同名文件，下载文件将覆盖本地文件。如果设置为错误，可避免本地系统上的现有文件被覆盖，并且操作模块会因下载错误而失败。 例如，Error: S3Download: File already exists and "overwrite" property for "destination" file is set to false. Cannot download.	布尔值	否	true

> **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 Amazon S3 对象复制到本地文件

以下示例说明了如何将 Amazon S3 对象复制到本地文件。

```
- name: DownloadMyFile
  action: S3Download
  inputs:
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
```

输入示例：将具有键前缀的 Amazon S3 存储桶中的所有 Amazon S3 对象复制到本地文件夹

下面的示例展示了如何将 Amazon S3 存储桶中带有键前缀的所有 Amazon S3 对象复制到本地文件夹。Amazon S3 没有文件夹概念，因此，将复制与键前缀匹配的所有对象。最大可下载数量为 1000。

```
- name: MyS3DownloadKeyprefix
  action: S3Download
  maxAttempts: 3
  inputs:
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
```

Output

无。

S3Upload（Linux、Windows、macOS）

使用 S3Upload 操作模块可以将文件从源文件/文件夹上传到某个 Amazon S3 位置。您可以在源位置指定的路径中使用通配符 (*)，以上传路径与通配符模式匹配的所有文件。

如果递归 S3Upload 操作失败，已上传的任何文件都将保留在目标 Amazon S3 存储桶中。

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 Amazon S3 键前缀。
- 将本地文件夹（必须将 `recurse` 设置为 `true`）复制到 Amazon S3 键前缀。

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 策略附加到与实例配置文件关联的 IAM 角色。该策略必须向目标 Amazon S3 存储桶授予 `s3:PutObject` 权限。例如，`arn:aws:s3:::BucketName/*`）。

Input

Key	说明	Type	必需	默认
`source`	files/folders 源所在的本地路径。`source` 支持星号通配符 (*)。	字符串	是	不适用
`destination`	上传源文件/文件夹的目标 Amazon S3 存储桶的路径。	字符串	是	不适用
`recurse`	如果设置为 `true`，则递归执行 S3Upload。	字符串	否	`false`

Key	说明	Type	必需	默认
expectedBucketOwner	目标路径中指定的 Amazon S3 存储桶的预期所有者账户 ID。我们建议您验证目标中指定的 Amazon S3 存储桶的所有权。	字符串	否	不适用

输入示例：将本地文件复制到 Amazon 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://amzn-s3-demo-destination-bucket/path/to/package.zip
      expectedBucketOwner: 123456789022
```

输入示例：将具有键前缀的 Amazon S3 存储桶中的所有文件复制到本地文件夹

以下示例展示了如何将本地文件夹中的所有文件复制到带有键前缀的 Amazon S3 存储桶中。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://amzn-s3-demo-destination-bucket/path/to/
      expectedBucketOwner: 123456789022
```

输入示例：将所有文件和文件夹从本地文件夹递归复制到 Amazon S3 存储桶

以下示例展示了如何将所有文件和文件夹从本地文件夹递归复制到带有键前缀的 Amazon S3 存储桶中。

```
- name: MyS3UploadFolder
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\*
      destination: s3://amzn-s3-demo-destination-bucket/path/to/
      recurse: true
      expectedBucketOwner: 123456789022
```

Output

无。

WebDownload（Linux、Windows、macOS）

WebDownload操作模块允许您通过 HTTP/HTTPS 协议从远程位置下载文件和资源（建议使用 HTTPS）。对下载数量或大小没有限制。该模块处理重试和指数回退逻辑。

根据用户输入，每次下载操作最多可尝试 5 次。这些尝试与 `steps` 文档的 `maxAttempts` 字段中指定的尝试不同，而是与操作模块失败有关。

此操作模块隐式处理重定向。除 `200` 外，所有 HTTP 状态代码都会引发错误。

Input

键名称	说明	Type	必需	默认
`source`	有效的 HTTP/HTTPS 网址（建议使用 HTTPS），它符合 RFC 3986 标准。允许使用链式表达式。	字符串	是	不适用
`destination`	本地系统上的绝对或相对文件或文件夹路径。文	字符串	是	不适用

键名称	说明	Type	必需	默认
	件夹路径必须以 / 结尾。如果文件夹路径不以 / 结尾，将被视为文件路径。该模块会创建成功下载所需的任何文件或文件夹。允许使用链式表达式。			
overwrite	启用后，下载的文件或资源将覆盖本地系统上的任何现有文件。如果未启用，则不会覆盖本地系统上的任何现有文件，并且操作模块会因错误而失败。启用覆盖并指定了校验和及算法后，只有在任何先前存在的文件的校验和与哈希值不匹配时，操作模块才会下载文件。	布尔值	否	true

键名称	说明	Type	必需	默认
checksum	当您指定校验和时，校验和会与使用提供的算法生成的已下载文件的哈希值进行比对。要启用文件验证，必须同时提供校验和与算法。允许使用链式表达式。	字符串	否	不适用
algorithm	用于计算校验和的算法。选项包括MD5、SHA1 SHA256、和 SHA512。要启用文件验证，必须同时提供校验和与算法。允许使用链式表达式。	字符串	否	不适用
ignoreCertificateErrors	启用后，SSL 证书验证会被忽略。	布尔值	否	false

Output

键名称	说明	Type
destination	以换行符分隔的字符串，指定了存储已下载文件或资	字符串

键名称	说明	Type				
	源的目标路径。					

输入示例：将远程文件下载到本地目标

```
  - name: DownloadRemoteFile
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: C:\testfolder\package.zip
```

输出：

```
{
 "destination": "C:\\testfolder\\package.zip"
}
```

输入示例：将多个远程文件下载到多个本地目标

```
  - name: DownloadRemoteFiles
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: /tmp/java14_renamed.zip
      - source: https://testdomain/path/to/java14.zip
        destination: /tmp/create_new_folder_and_add_java14_as_zip/
```

输出：

```
{
 "destination": "/tmp/create_new_folder/java14_renamed.zip\n/tmp/
create_new_folder_and_add_java14_as_zip/java14.zip"
}
```

输入示例：下载一个远程文件而不覆盖本地目标，然后下载另一个带有文件验证功能的远程文件

```
  - name: DownloadRemoteMultipleProperties
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: C:\create_new_folder\java14_renamed.zip
        overwrite: false
      - source: https://testdomain/path/to/java14.zip
        destination: C:\create_new_folder_and_add_java14_as_zip\
        checksum: ac68bbf921d953d1cfab916cb6120864
        algorithm: MD5
        overwrite: true
```

输出：

```
{
 "destination": "C:\\create_new_folder\\java14_renamed.zip\nC:\
\create_new_folder_and_add_java14_as_zip\\java14.zip"
}
```

输入示例：下载远程文件并忽略 SSL 认证验证

```
  - name: DownloadRemoteIgnoreValidation
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://www.bad-ssl.com/resource
        destination: /tmp/downloads/
        ignoreCertificateErrors: true
```

输出：

```
{
 "destination": "/tmp/downloads/resource"
}
```

文件系统操作模块

下一节详细介绍了执行文件系统操作的操作模块。

文件系统操作操作模块

- AppendFile（Linux、Windows、macOS）
- CopyFile（Linux、Windows、macOS）
- CopyFolder（Linux、Windows、macOS）
- CreateFile（Linux、Windows、macOS）
- CreateFolder（Linux、Windows、macOS）
- CreateSymlink（Linux、Windows、macOS）
- DeleteFile（Linux、Windows、macOS）
- DeleteFolder（Linux、Windows、macOS）
- ListFiles（Linux、Windows、macOS）
- MoveFile（Linux、Windows、macOS）
- MoveFolder（Linux、Windows、macOS）
- ReadFile（Linux、Windows、macOS）
- SetFileEncoding（Linux、Windows、macOS）
- SetFileOwner（Linux、Windows、macOS）
- SetFolderOwner（Linux、Windows、macOS）
- SetFilePermissions（Linux、Windows、macOS）
- SetFolderPermissions（Linux、Windows、macOS）

AppendFile（Linux、Windows、macOS）

AppendFile操作模块将指定内容添加到文件中先前存在的内容中。

如果文件编码值与默认编码 (`utf-8`) 值不同，可以使用 `encoding` 选项指定文件编码值。默认情况下，`utf-16` 和 `utf-32` 使用小端字节序编码。

发生以下情况时，操作模块会返回错误：

- 指定的文件在运行时不存在。
- 您没有修改文件内容的写入权限。
- 该模块在文件操作期间遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件路径。	字符串	是	不适用	不适用	是
content	要附加到文件的内容。	字符串	否	空字符串	不适用	是
encoding	编码标准。	字符串	否	utf8	utf8、utf-LE、utf-16 LE 、utf16-BE、utf-16 BE 、utf32、 LE、utf-32 LE 、utf32-BE 和 utf-32-BE 。编码选项的值不区分大小写。	是

输入示例：附加文件而不编码（Linux）

```
- name: AppendingFileWithOutEncodingLinux
  action: AppendFile
  inputs:
    - path: ./Sample.txt
      content: "The string to be appended to the file"
```

输入示例：附加文件而不编码（Windows）

```
- name: AppendingFileWithOutEncodingWindows
  action: AppendFile
```

```
  inputs:
    - path: C:\MyFolder\MyFile.txt
      content: "The string to be appended to the file"
```

输入示例：附加文件并编码（Linux）

```
- name: AppendingFileWithEncodingLinux
  action: AppendFile
  inputs:
    - path: /FolderName/SampleFile.txt
      content: "The string to be appended to the file"
      encoding: UTF-32
```

输入示例：附加文件并编码（Windows）

```
- name: AppendingFileWithEncodingWindows
  action: AppendFile
  inputs:
    - path: C:\MyFolderName\SampleFile.txt
      content: "The string to be appended to the file"
      encoding: UTF-32
```

输入示例：以空字符串附加文件（Linux）

```
- name: AppendingEmptyStringLinux
  action: AppendFile
  inputs:
    - path: /FolderName/SampleFile.txt
```

输入示例：以空字符串附加文件（Windows）

```
- name: AppendingEmptyStringWindows
  action: AppendFile
  inputs:
    - path: C:\MyFolderName\SampleFile.txt
```

Output

无。

CopyFile（Linux、Windows、macOS）

CopyFile操作模块将文件从指定源复制到指定目标。默认情况下，如果目标文件夹在运行时不存在，该模块将会以递归方式创建该文件夹。

如果指定文件夹中已存在具有指定名称的文件，则默认情况下，操作模块将覆盖现有文件。您可以将覆盖选项设置为 `false`，覆盖这一默认行为。当覆盖选项设置为 `false`，并且在指定位置已经存在具有指定名称的文件时，操作模块将返回错误。此选项的工作原理与 Linux 中的 `cp` 命令相同，默认情况下会覆盖。

源文件名可以包含通配符 (*)。只有在最后一个文件路径分隔符（/ 或 \）之后才可使用通配符。如果源文件名中包含通配符，则所有与通配符匹配的文件都将复制到目标文件夹。如果要使用通配符移动多个文件，则 `destination` 选项的输入必须以文件路径分隔符（/ 或 \）结尾，这表示目标输入是一个文件夹。

如果目标文件名与源文件名不同，则可以使用 `destination` 选项指定目标文件名。如果未指定目标文件名，则使用源文件的名称来创建目标文件。最后一个文件路径分隔符（/ 或 \）之后的任何文本都将视为文件名。如果要使用与源文件相同的文件名，则 `destination` 选项的输入必须以文件路径分隔符（/ 或 \）结尾。

发生以下情况时，操作模块会返回错误：

- 您无权在指定文件夹中创建文件。
- 源文件在运行时不存在。
- 已存在具有指定文件名的文件夹，且 `overwrite` 选项设置为 `false`。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`source`	源文件路径。	字符串	是	不适用	不适用	是
`destination`	目标文件路径。	字符串	是	不适用	不适用	是
`overwrite`	如果设置为错误，当指	布尔值	否	`true`	不适用	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
	定位置已经存在具有指定名称的文件时，目标文件将不会被替换。					

输入示例：复制文件（Linux）

```
- name: CopyingAFileLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
```

输入示例：复制文件（Windows）

```
- name: CopyingAFileWindows
  action: CopyFile
  inputs:
    - source: C:\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
```

输入示例：使用源文件名复制文件（Linux）

```
- name: CopyingFileWithSourceFileNameLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/
```

输入示例：使用源文件名复制文件（Windows）

```
- name: CopyingFileWithSourceFileNameWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
```

```
        destination: C:\MyFolder\
```

输入示例：使用通配符复制文件（Linux）

```
- name: CopyingFilesWithWildCardLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

输入示例：使用通配符复制文件（Windows）

```
- name: CopyingFilesWithWildCardWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

输入示例：复制文件而不覆盖（Linux）

```
- name: CopyingFilesWithoutOverwriteLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
      overwrite: false
```

输入示例：复制文件而不覆盖（Windows）

```
- name: CopyingFilesWithoutOverwriteWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
      overwrite: false
```

Output

无。

CopyFolder（Linux、Windows、macOS）

CopyFolder操作模块将文件夹从指定源复制到指定目标。source 选项的输入为要复制的文件夹，destination 选项的输入为源文件夹内容被复制的文件夹。默认情况下，如果目标文件夹在运行时不存在，该模块将会以递归方式创建该文件夹。

如果指定文件夹中已经存在具有指定名称的文件夹，则默认情况下，操作模块会覆盖现有文件夹。您可以将覆盖选项设置为 false，覆盖这一默认行为。如果将覆盖选项设置为 false，并且在指定位置已经存在具有指定名称的文件夹，操作模块将返回错误。

源文件夹名称可以包含通配符 (*)。只有在最后一个文件路径分隔符（/ 或 \）之后才可使用通配符。如果源文件夹名称中包含通配符，则所有与通配符匹配的文件夹都将复制到目标文件夹。如果要使用通配符复制多个文件夹，则 destination 选项的输入必须以文件路径分隔符（/ 或 \）结尾，这表示目标输入是一个文件夹。

如果目标文件夹名称与源文件夹名称不同，则可以使用 destination 选项指定目标文件夹名称。如果未指定目标文件夹名称，则使用源文件夹的名称来创建目标文件夹。最后一个文件路径分隔符（/ 或 \）之后的任何文本都将视为文件夹名称。如果要使用与源文件夹相同的文件夹名称，则 destination 选项的输入必须以文件路径分隔符（/ 或 \）结尾。

发生以下情况时，操作模块会返回错误：

- 您无权在指定文件夹中创建文件夹。
- 源文件夹在运行时不存在。
- 已存在具有指定文件夹名称的文件夹，且 overwrite 选项设置为 false。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
source	源文件夹路径。	字符串	是	不适用	不适用	是
destination	目标文件夹路径。	字符串	是	不适用	不适用	是
overwrite	如果设置为错误，当指	布尔值	否	true	不适用	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
	定位置中已经存在具有指定名称的文件夹时，目标文件夹将不会被替换。					

输入示例：复制文件夹（Linux）

```
- name: CopyingAFolderLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
      destination: /MyFolder/destinationFolder
```

输入示例：复制文件夹（Windows）

```
- name: CopyingAFolderWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\destinationFolder
```

输入示例：使用源文件夹名称复制文件夹（Linux）

```
- name: CopyingFolderSourceFolderNameLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/
```

输入示例：使用源文件夹名称复制文件夹（Windows）

```
- name: CopyingFolderSourceFolderNameWindows
  action: CopyFolder
  inputs:
```

```
      - source: C:\Sample\MyFolder\SampleFolder
        destination: C:\MyFolder\
```

输入示例：使用通配符复制文件夹（Linux）

```
- name: CopyingFoldersWithWildCardLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

输入示例：使用通配符复制文件夹（Windows）

```
- name: CopyingFoldersWithWildCardWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

输入示例：在不覆盖的情况下复制文件夹（Linux）

```
- name: CopyingFoldersWithoutOverwriteLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/destinationFolder
      overwrite: false
```

输入示例：在不覆盖的情况下复制文件夹（Windows）

```
- name: CopyingFoldersWithoutOverwrite
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SourceFolder
      destination: C:\MyFolder\destinationFolder
      overwrite: false
```

Output

无。

CreateFile（Linux、Windows、macOS）

CreateFile操作模块在指定位置创建文件。默认情况下，如有需要，该模块还会以递归方式创建父文件夹。

如果指定文件夹中已存在该文件，则默认情况下，操作模块会截断或覆盖现有文件。您可以将覆盖选项设置为 `false`，覆盖这一默认行为。当覆盖选项设置为 `false`，并且在指定位置已经存在具有指定名称的文件时，操作模块将返回错误。

如果文件编码值与默认编码 (`utf-8`) 值不同，可以使用 `encoding` 选项指定文件编码值。默认情况下，`utf-16` 和 `utf-32` 使用小端字节序编码。

`owner`、`group`、和 `permissions` 是可选输入。`permissions` 的输入必须是字符串值。如果未提供默认值，将使用默认值创建文件。Windows 平台不支持这些选项。如果在 Windows 平台上使用选项 `owner`、`group` 和 `permissions`，则此操作模块将验证并返回错误。

此操作模块可以创建一个文件，其权限由操作系统的默认 `umask` 值定义。如果想覆盖默认值，则必须设置 `umask` 值。

发生以下情况时，操作模块会返回错误：

- 您无权在指定的父文件夹中创建文件或文件夹。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`path`	文件路径。	字符串	是	不适用	不适用	是
`content`	文件的文本内容。	字符串	否	不适用	不适用	是
`encoding`	编码标准。	字符串	否	`utf8`	`utf8`、`utf-LE`、`utf-16LE`、`utf16-BE`、`utf-16BE`	是 tf-16、

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
					、utf32、u LE、utf-32 LE 、utf32-BE 和 utf-32-BE 。编码选项的值不区分大小写。	32-
owner	用户名或 ID。	字符串	否	不适用	不适用	Windows 中不支持。
group	组名称或 ID。	字符串	否	当前用户。	不适用	Windows 中不支持。
permissions	文件权限。	字符串	否	0666	不适用	Windows 中不支持。
overwrite	如果指定文件的名称已经存在，则默认情况下，将此值设置为 false 可防止文件被截断或覆盖。	布尔值	否	true	不适用	是

输入示例：创建文件而不覆盖（Linux）

```
- name: CreatingFileWithoutOverwriteLinux
  action: CreateFile
  inputs:
```

```
      - path: /home/UserName/Sample.txt
        content: The text content of the sample file.
        overwrite: false
```

输入示例：创建文件而不覆盖（Windows）

```
- name: CreatingFileWithoutOverwriteWindows
  action: CreateFile
  inputs:
    - path: C:\Temp\Sample.txt
      content: The text content of the sample file.
      overwrite: false
```

输入示例：创建带文件属性的文件

```
- name: CreatingFileWithFileProperties
  action: CreateFile
  inputs:
    - path: SampleFolder/Sample.txt
      content: The text content of the sample file.
      encoding: UTF-16
      owner: Ubuntu
      group: UbuntuGroup
      permissions: 0777
   - path: SampleFolder/SampleFile.txt
      permissions: 755
    - path: SampleFolder/TextFile.txt
      encoding: UTF-16
      owner: root
      group: rootUserGroup
```

输入示例：创建不带文件属性的文件

```
- name: CreatingFileWithoutFileProperties
  action: CreateFile
  inputs:
    - path: ./Sample.txt
    - path: Sample1.txt
```

输入示例：创建一个空文件以跳过 Linux 清理脚本中的某一部分

```
- name: CreateSkipCleanupfile
```

```
    action: CreateFile
    inputs:
      - path: <skip section file name>
```

有关更多信息，请参阅 覆盖 Linux 清理脚本

Output

无。

CreateFolder（Linux、Windows、macOS）

CreateFolder操作模块在指定位置创建一个文件夹。默认情况下，如有需要，该模块还会以递归方式创建父文件夹。

如果指定文件夹中已存在该文件夹，则默认情况下，操作模块会截断或覆盖现有文件夹。您可以将覆盖选项设置为 `false`，覆盖这一默认行为。如果将覆盖选项设置为 `false`，并且在指定位置已经存在具有指定名称的文件夹，操作模块将返回错误。

`owner`、`group`、和 `permissions` 是可选输入。`permissions` 的输入必须是字符串值。Windows 平台不支持这些选项。如果在 Windows 平台上使用选项 `owner`、`group` 和 `permissions`，则此操作模块将验证并返回错误。

此操作模块可以创建一个文件夹，其权限由操作系统的默认 umask 值定义。如果想覆盖默认值，则必须设置 umask 值。

发生以下情况时，操作模块会返回错误：

- 您无权在指定位置创建文件夹。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`path`	文件夹路径。	字符串	是	不适用	不适用	是
`owner`	用户名或 ID。	字符串	否	当前用户。	不适用	Windows 中不支持。

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
group	组名称或 ID。	字符串	否	当前用户的组。	不适用	Windows 中不支持。
permissions	文件夹权限。	字符串	否	0777	不适用	Windows 中不支持。
overwrite	如果指定文件的名称已经存在，则默认情况下，将此值设置为 false 可防止文件被截断或覆盖。	布尔值	否	true	不适用	是

输入示例：创建文件夹（Linux）

```
- name: CreatingFolderLinux
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/
```

输入示例：创建文件夹（Windows）

```
- name: CreatingFolderWindows
  action: CreateFolder
  inputs:
    - path: C:\MyFolder
```

输入示例：创建指定文件夹属性的文件夹

```
- name: CreatingFolderWithFolderProperties
  action: CreateFolder
  inputs:
```

```
      - path: /Sample/MyFolder/Sample/
        owner: SampleOwnerName
        group: SampleGroupName
        permissions: 0777
      - path: /Sample/MyFolder/SampleFoler/
        permissions: 777
```

输入示例：创建一个覆盖现有文件夹的文件夹（如有）。

```
- name: CreatingFolderWithOverwrite
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      overwrite: true
```

Output

无。

CreateSymlink（Linux、Windows、macOS）

CreateSymlink操作模块创建符号链接或包含对另一个文件的引用的文件。Windows 平台不支持此模块。

path 和 target 选项的输入可以是绝对路径，也可以是相对路径。如果 path 选项的输入是相对路径，则在创建链接时它将替换为绝对路径。

默认情况下，当指定文件夹中已存在具有指定名称的链接时，操作模块会返回错误。您可以将 force 选项设置为 true，覆盖这一默认行为。当 force 选项设置为 true 时，模块将覆盖现有链接。

如果父文件夹不存在，则默认情况下，操作模块会以递归方式创建该文件夹。

发生以下情况时，操作模块会返回错误：

- 目标文件在运行时不存在。
- 已存在具有指定名称的非符号链接文件。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件路径。	字符串	是	不适用	不适用	Windows 中不支持。
target	符号链接指向的目标文件路径。	字符串	是	不适用	不适用	Windows 中不支持。
force	当名称相同的链接存在时，将强制创建链接。	布尔值	否	false	不适用	Windows 中不支持。

输入示例：创建强制创建链接的符号链接

```
- name: CreatingSymbolicLinkWithForce
  action: CreateSymlink
  inputs:
    - path: /Folder2/Symboliclink.txt
      target: /Folder/Sample.txt
      force: true
```

输入示例：创建不强制创建链接的符号链接

```
- name: CreatingSymbolicLinkWithOutForce
  action: CreateSymlink
  inputs:
    - path: Symboliclink.txt
      target: /Folder/Sample.txt
```

Output

无。

DeleteFile（Linux、Windows、macOS）

DeleteFile操作模块删除指定位置的一个或多个文件。

path 的输入应该是有效的文件路径或文件名中带有通配符 (*) 的文件路径。如果在文件名中已指定通配符，则同一文件夹中与通配符匹配的所有文件都将被删除。

发生以下情况时，操作模块会返回错误：

- 您无权执行删除的操作。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件路径。	字符串	是	不适用	不适用	是

输入示例：删除单个文件（Linux）

```
- name: DeletingSingleFileLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/Sample.txt
```

输入示例：删除单个文件（Windows）

```
- name: DeletingSingleFileWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\Sample.txt
```

输入示例：删除以“日志”结尾的文件（Linux）

```
- name: DeletingFileEndingWithLogLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/*log
```

输入示例：删除以“日志”结尾的文件（Windows）

```
- name: DeletingFileEndingWithLogWindows
  action: DeleteFile
```

```
    inputs:
      - path: C:\SampleFolder\MyFolder\*log
```

输入示例：删除指定文件夹中的所有文件（Linux）

```
- name: DeletingAllFilesInAFolderLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/*
```

输入示例：删除指定文件夹中的所有文件（Windows）

```
- name: DeletingAllFilesInAFolderWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\*
```

Output

无。

DeleteFolder（Linux、Windows、macOS）

DeleteFolder操作模块删除文件夹。

如果该文件夹不为空，则必须将 `force` 选项设置为 `true` 才能删除该文件夹及其内容。如果您未将 `force` 选项设置为 `true`，并且您要删除的文件夹不为空，则操作模块会返回错误。`force` 选项的默认值为 `false`。

发生以下情况时，操作模块会返回错误：

- 您无权执行删除的操作。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`path`	文件夹路径。	字符串	是	不适用	不适用	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
force	无论文件夹是否为空，都将删除该文件夹。	布尔值	否	false	不适用	是

输入示例：使用 **force** 选项删除非空文件夹（Linux）

```
- name: DeletingFolderWithForceOptionLinux
  action: DeleteFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      force: true
```

输入示例：使用 **force** 选项删除非空文件夹（Windows）

```
- name: DeletingFolderWithForceOptionWindows
  action: DeleteFolder
  inputs:
    - path: C:\Sample\MyFolder\Sample\
      force: true
```

输入示例：删除文件夹（Linux）

```
- name: DeletingFolderWithOutForceLinux
  action: DeleteFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
```

输入示例：删除文件夹（Windows）

```
- name: DeletingFolderWithOutForce
  action: DeleteFolder
  inputs:
    - path: C:\Sample\MyFolder\Sample\
```

Output

无。

ListFiles（Linux、Windows、macOS）

ListFiles操作模块列出了指定文件夹中的文件。当递归选项设置为 `true` 时，将列出子文件夹中的文件。默认情况下，此模块不列出子文件夹中的文件。

要列出名称与指定模式匹配的所有文件，请使用 `fileNamePattern` 选项提供该模式。`fileNamePattern` 选项接受通配符 (*) 值。提供 `fileNamePattern` 时，将返回与指定文件名格式匹配的所有文件。

发生以下情况时，操作模块会返回错误：

- 指定的文件夹在运行时不存在。
- 您无权在指定的父文件夹中创建文件或文件夹。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`path`	文件夹路径。	字符串	是	不适用	不适用	是
`fileNamePattern`	要匹配以列出名称与该模式匹配的所有文件的模式。	字符串	否	不适用	不适用	是
`recursive`	递归列出文件夹中的文件。	布尔值	否	`false`	不适用	是

输入示例：列出指定文件夹中的文件（Linux）

```
- name: ListingFilesInSampleFolderLinux
  action: ListFiles
  inputs:
```

```
      - path: /Sample/MyFolder/Sample
```

输入示例：列出指定文件夹中的文件（Windows）

```
- name: ListingFilesInSampleFolderWindows
  action: ListFiles
  inputs:
    - path: C:\Sample\MyFolder\Sample
```

输入示例：列出以“日志”结尾的文件（Linux）

```
- name: ListingFilesWithEndingWithLogLinux
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/
      fileNamePattern: *log
```

输入示例：列出以“日志”结尾的文件（Windows）

```
- name: ListingFilesWithEndingWithLogWindows
  action: ListFiles
  inputs:
    - path: C:\Sample\MyFolder\
      fileNamePattern: *log
```

输入示例：递归列出文件

```
- name: ListingFilesRecursively
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/
      recursive: true
```

Output

键名称	说明	Type
files	文件列表。	字符串

输出示例

```
{
 "files": "/sample1.txt,/sample2.txt,/sample3.txt"
}
```

MoveFile（Linux、Windows、macOS）

MoveFile操作模块将文件从指定源移动到指定目标。

如果指定文件夹中已存在该文件，则默认情况下，操作模块会覆盖现有文件。您可以将覆盖选项设置为 `false`，覆盖这一默认行为。当覆盖选项设置为 `false`，并且在指定位置已经存在具有指定名称的文件时，操作模块将返回错误。此选项的工作原理与 Linux 中的 `mv` 命令相同，默认情况下会覆盖。

源文件名可以包含通配符 (*)。只有在最后一个文件路径分隔符（/ 或 \）之后才可使用通配符。如果源文件名中包含通配符，则所有与通配符匹配的文件都将复制到目标文件夹。如果要使用通配符移动多个文件，则 `destination` 选项的输入必须以文件路径分隔符（/ 或 \）结尾，这表示目标输入是一个文件夹。

如果目标文件名与源文件名不同，则可以使用 `destination` 选项指定目标文件名。如果未指定目标文件名，则使用源文件的名称来创建目标文件。最后一个文件路径分隔符（/ 或 \）之后的任何文本都将视为文件名。如果要使用与源文件相同的文件名，则 `destination` 选项的输入必须以文件路径分隔符（/ 或 \）结尾。

发生以下情况时，操作模块会返回错误：

- 您无权在指定文件夹中创建文件。
- 源文件在运行时不存在。
- 已存在具有指定文件名的文件夹，且 `overwrite` 选项设置为 `false`。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`source`	源文件路径。	字符串	是	不适用	不适用	是
`destination`	目标文件路径。	字符串	是	不适用	不适用	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
overwrite	如果设置为错误，当指定位置已经存在具有指定名称的文件时，目标文件将不会被替换。	布尔值	否	true	不适用	是

输入示例：移动文件（Linux）

```
- name: MovingAFileLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
```

输入示例：移动文件（Windows）

```
- name: MovingAFileWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
```

输入示例：使用源文件名移动文件（Linux）

```
- name: MovingFileWithSourceFileNameLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/
```

输入示例：使用源文件名移动文件（Windows）

```
- name: MovingFileWithSourceFileNameWindows
  action: MoveFile
```

```
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder
```

输入示例：使用通配符移动文件（Linux）

```
- name: MovingFilesWithWildCardLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

输入示例：使用通配符移动文件（Windows）

```
- name: MovingFilesWithWildCardWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder
```

输入示例：移动文件而不覆盖（Linux）

```
- name: MovingFilesWithoutOverwriteLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
      overwrite: false
```

输入示例：移动文件而不覆盖（Windows）

```
- name: MovingFilesWithoutOverwrite
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
      overwrite: false
```

Output

无。

MoveFolder（Linux、Windows、macOS）

MoveFolder操作模块将文件夹从指定源移动到指定目标。source 选项的输入是要移动的文件夹，而 destination 选项的输入是源文件夹内容被移动的文件夹。

如果目标父文件夹或 destination 选项的输入在运行时不存在，则默认情况下，模块会在指定的目标上递归创建文件夹。

如果目标文件夹中已存在与源文件夹相同的文件夹，则默认情况下，操作模块会覆盖现有文件夹。您可以将覆盖选项设置为 false，覆盖这一默认行为。如果将覆盖选项设置为 false，并且在指定位置已经存在具有指定名称的文件夹，操作模块将返回错误。

源文件夹名称可以包含通配符 (*)。只有在最后一个文件路径分隔符（/ 或 \）之后才可使用通配符。如果源文件夹名称中包含通配符，则所有与通配符匹配的文件夹都将复制到目标文件夹。如果要使用通配符移动多个文件夹，则 destination 选项的输入必须以文件路径分隔符（/ 或 \）结尾，这表示目标输入是一个文件夹。

如果目标文件夹名称与源文件夹名称不同，则可以使用 destination 选项指定目标文件夹名称。如果未指定目标文件夹名称，则使用源文件夹的名称来创建目标文件夹。最后一个文件路径分隔符（/ 或 \）之后的任何文本都将视为文件夹名称。如果要使用与源文件夹相同的文件夹名称，则 destination 选项的输入必须以文件路径分隔符（/ 或 \）结尾。

发生以下情况时，操作模块会返回错误：

- 您无权在目标文件夹中创建文件夹。
- 源文件夹在运行时不存在。
- 已存在具有指定名称的文件夹，且 overwrite 选项设置为 false。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
source	源文件夹路径。	字符串	是	不适用	不适用	是
destination	目标文件夹路径。	字符串	是	不适用	不适用	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
overwrite	如果设置为错误，当指定位置中已经存在具有指定名称的文件夹时，目标文件夹将不会被替换。	布尔值	否	true	不适用	是

输入示例：移动文件夹（Linux）

```
- name: MovingAFolderLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/destinationFolder
```

输入示例：移动文件夹（Windows）

```
- name: MovingAFolderWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\SourceFolder
      destination: C:\MyFolder\destinationFolder
```

输入示例：使用源文件夹名称移动文件夹（Linux）

```
- name: MovingFolderWithSourceFolderNameLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
      destination: /MyFolder/
```

输入示例：使用源文件夹名称移动文件夹（Windows）

```
- name: MovingFolderWithSourceFolderNameWindows
```

```
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\
```

输入示例：使用通配符移动文件夹（Linux）

```
- name: MovingFoldersWithWildCardLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

输入示例：使用通配符移动文件夹（Windows）

```
- name: MovingFoldersWithWildCardWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

输入示例：在不覆盖的情况下移动文件夹（Linux）

```
- name: MovingFoldersWithoutOverwriteLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
  destination: /MyFolder/destinationFolder
  overwrite: false
```

输入示例：在不覆盖的情况下移动文件夹（Windows）

```
- name: MovingFoldersWithoutOverwriteWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\destinationFolder
      overwrite: false
```

Output

无。

ReadFile（Linux、Windows、macOS）

ReadFile操作模块读取字符串类型的文本文件的内容。该模块可用于读取文件内容，以便通过链接在后续步骤中使用，或者用于读取数据到 `console.log` 文件。如果指定的路径是符号链接，则此模块将返回目标文件的内容。该模块仅支持文本文件。

如果文件编码值与默认编码 (`utf-8`) 值不同，可以使用 `encoding` 选项指定文件编码值。默认情况下，`utf-16` 和 `utf-32` 使用小端字节序编码。

默认情况下，此模块无法将文件内容打印到 `console.log` 文件中。您可以通过将 `printFileContent` 属性设置为 `true` 来覆盖此设置。

该模块只能返回文件的内容。该模块无法解析文件（例如 Excel 或 JSON 文件）。

发生以下情况时，操作模块会返回错误：

- 该文件在运行时不存在。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
`path`	文件路径。	字符串	是	不适用	不适用	是
`encoding`	编码标准。	字符串	否	`utf8`	`utf8`、`utf-LE`、`utf-16LE`、`utf16-BE`、`utf-16BE`、`utf32`、`utf-32LE`、`utf32-BE` 和 `utf-32-BE`。编码选项的值不	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
					区分大小写。	
printFileContent	将文件内容打印到 console.log 文件中。	布尔值	否	false	不适用	可以。

输入示例：读取文件（Linux）

```
- name: ReadingFileLinux
  action: ReadFile
  inputs:
    - path: /home/UserName/SampleFile.txt
```

输入示例：读取文件（Windows）

```
- name: ReadingFileWindows
  action: ReadFile
  inputs:
    - path: C:\Windows\WindowsUpdate.log
```

输入示例：读取文件并指定编码标准

```
- name: ReadingFileWithFileEncoding
  action: ReadFile
  inputs:
    - path: /FolderName/SampleFile.txt
      encoding: UTF-32
```

输入示例：读取文件并将其打印到 **console.log** 文件

```
- name: ReadingFileToConsole
  action: ReadFile
  inputs:
    - path: /home/UserName/SampleFile.txt
```

```
        printFileContent: true
```

Output

字段	描述	Type
content	文件内容。	字符串

输出示例

```
{
 "content" : "The file content"
}
```

SetFileEncoding（Linux、Windows、macOS）

SetFileEncoding操作模块修改现有文件的编码属性。该模块可以将文件编码从 utf-8 转换为指定的编码标准。默认情况下，utf-16 和 utf-32 为小端字节序编码。

发生以下情况时，操作模块会返回错误：

- 您无权执行指定修改。
- 该文件在运行时不存在。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件路径。	字符串	是	不适用	不适用	是
encoding	编码标准。	字符串	否	utf8	utf8、utf-LE、utf-16LE 、utf16-BE、utf-16BE 、utf32、	是

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
					LE、utf-32 LE 、utf32-BE 和 utf-32-BE 。编码选项的值不区分大小写。	

输入示例：设置文件编码属性

```
- name: SettingFileEncodingProperty
  action: SetFileEncoding
  inputs:
    - path: /home/UserName/SampleFile.txt
      encoding: UTF-16
```

Output

无。

SetFileOwner（Linux、Windows、macOS）

SetFileOwner操作模块修改现有文件的owner和group所有者属性。如果指定文件是符号链接，则模块将修改源文件的 owner 属性。Windows 平台不支持此模块。

该模块接受用户名和组名作为输入。如果未提供组名，则该模块会将文件的组所有者分配给该用户所属的组。

发生以下情况时，操作模块会返回错误：

- 您无权执行指定修改。
- 指定的用户名或组名在运行时不存在。
- 该文件在运行时不存在。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件路径。	字符串	是	不适用	不适用	Windows 中不支持。
owner	用户名。	字符串	是	不适用	不适用	Windows 中不支持。
group	用户组的名称。	字符串	否	用户属于的组的名称。	不适用	Windows 中不支持。

输入示例：设置文件所有者属性而不指定用户组名称

```
- name: SettingFileOwnerPropertyNoGroup
  action: SetFileOwner
  inputs:
    - path: /home/UserName/SampleText.txt
      owner: LinuxUser
```

输入示例：通过指定所有者和用户组来设置文件所有者属性

```
- name: SettingFileOwnerProperty
  action: SetFileOwner
  inputs:
    - path: /home/UserName/SampleText.txt
      owner: LinuxUser
      group: LinuxUserGroup
```

Output

无。

SetFolderOwner（Linux、Windows、macOS）

SetFolderOwner操作模块以递归方式修改现有文件夹的owner和group所有者属性。默认情况下，该模块可以修改文件夹中所有内容的所有权。您可以将 recursive 选项设置为 false 以覆盖此行为。Windows 平台不支持此模块。

该模块接受用户名和组名作为输入。如果未提供组名，则该模块会将文件的组所有者分配给该用户所属的组。

发生以下情况时，操作模块会返回错误：

- 您无权执行指定修改。
- 指定的用户名或组名在运行时不存在。
- 该文件夹在运行时不存在。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件夹路径。	字符串	是	不适用	不适用	Windows 中不支持。
owner	用户名。	字符串	是	不适用	不适用	Windows 中不支持。
group	用户组的名称。	字符串	否	用户属于的组的名称。	不适用	Windows 中不支持。
recursive	如果设置为 false，将覆盖修改文件夹所有内容所有权的默认行为。	布尔值	否	true	不适用	Windows 中不支持。

输入示例：设置文件夹所有者属性而不指定用户组名称

```
- name: SettingFolderPropertyWithOutGroup
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
```

输入示例：设置文件夹所有者属性而不覆盖文件夹中所有内容的所有权

```
- name: SettingFolderPropertyWithOutRecursively
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
      recursive: false
```

输入示例：通过指定用户组名称来设置文件所有权属性

```
- name: SettingFolderPropertyWithGroup
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
      group: LinuxUserGroup
```

Output

无。

SetFilePermissions（Linux、Windows、macOS）

SetFilePermissions操作模块修改现有文件的。`permissions`Windows 平台不支持此模块。

`permissions` 的输入必须是字符串值。

此操作模块将创建一个文件，其权限由操作系统默认的 umask 值定义。如果想覆盖默认值，则必须设置 `umask` 值。

发生以下情况时，操作模块会返回错误：

- 您无权执行指定修改。
- 该文件在运行时不存在。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件路径。	字符串	是	不适用	不适用	Windows 中不支持。
permissions	文件权限。	字符串	是	不适用	不适用	Windows 中不支持。

输入示例：修改文件权限

```
- name: ModifyingFilePermissions
  action: SetFilePermissions
  inputs:
    - path: /home/UserName/SampleFile.txt
      permissions: 766
```

Output

无。

SetFolderPermissions（Linux、Windows、macOS）

SetFolderPermissions操作模块以递归方式修改现有文件夹及其所有子文件和子文件夹。permissions默认情况下，此模块可以修改指定文件夹中所有内容的权限。您可以将 recursive 选项设置为 false 以覆盖此行为。Windows 平台不支持此模块。

permissions 的输入必须是字符串值。

此操作模块可以根据操作系统的默认 umask 值修改权限。如果想覆盖默认值，则必须设置 umask 值。

发生以下情况时，操作模块会返回错误：

- 您无权执行指定修改。
- 该文件夹在运行时不存在。
- 操作模块在执行操作时遇到错误。

Input

键名称	说明	Type	必需	默认 值	可接受值	支持全平台
path	文件夹路径。	字符串	是	不适用	不适用	Windows 中不支持。
permissions	文件夹权限。	字符串	是	不适用	不适用	Windows 中不支持。
recursive	如果设置为 false，将覆盖修改文件夹所有内容权限的默认行为。	布尔值	否	true	不适用	Windows 中不支持。

输入示例：设置文件夹权限

```
- name: SettingFolderPermissions
  action: SetFolderPermissions
  inputs:
    - path: SampleFolder/
      permissions: 0777
```

输入示例：设置文件夹权限而不修改文件夹所有内容的权限

```
- name: SettingFolderPermissionsNoRecursive
  action: SetFolderPermissions
  inputs:
    - path: /home/UserName/SampleFolder/
      permissions: 777
      recursive: false
```

Output

无。

软件安装操作

下一部分介绍了安装或卸载软件的操作模块。

IAM 要求

如果您的安装下载路径是 S3 URI，则与您的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。要授予所需权限，请将 S3:GetObject IAM 策略附加到与实例配置文件关联的 IAM 角色，并指定存储桶的路径。例如，*arn:aws:s3:::**BucketName**/**）。

复杂 MSI 输入

如果您的输入字符串包含双引号字符 (")，则必须使用以下方法之一来确保其得到正确解释：

- 您可以在字符串的外部使用单引号 (') 来包含它，并在字符串内部使用双引号 (")，如下面的示例所示。

  ```
  properties:
    COMPANYNAME: '"Acme ""Widgets"" and ""Gizmos."""'
  ```

  在这种情况下，如果您需要在字符串中使用撇号，则必须对其进行转义。这意味着需要在撇号前使用另一个单引号 (')。
- 你可以在字符串的外面使用双引号 (") 来包含它。您可以使用反斜杠字符 (\) 对字符串中的任何双引号进行转义，如以下示例所示。

  ```
  properties:
    COMPANYNAME: "\"Acme \"\"Widgets\"\" and \"\"Gizmos.\"\"\""
  ```

这两种方法都将值 COMPANYNAME="Acme ""Widgets"" and ""Gizmos.""" 传递给 msiexec 命令。

软件安装操作模块

- InstallMSI (Windows)
- UninstallMSI (Windows)

InstallMSI (Windows)

InstallMSI 操作模块使用 MSI 文件安装 Windows 应用程序。您可以使用本地路径、S3 对象 URI 或 Web URL 来指定 MSI 文件。重新启动选项可配置系统的重新启动行为。

AWSTOE 根据操作模块的输入参数生成msiexec命令。path（MSI 文件位置）和 logFile（日志文件位置）输入参数的值必须用引号（"）括起来。

以下 MSI 退出代码视为成功：

- 0（成功）
- 1614（ERROR_PRODUCT_UNINSTALLED）
- 1641（已启动重启）
- 3010（需要重启）

Input

键名称	说明	Type	必需	默认 值	可接受值
path	使用以下之一指定 MSI 文件位置： • 本地文件路径。路径可以是绝对路径，也可以是相对路径 • 有效的 S3 对象 URI。 • 符合 RFC 3986 标准的有效 HTTP/HTTPS 网址（建议使用 HTTPS）。	字符串	是	不适用	不适用

键名称	说明	Type	必需	默认 值	可接受值
	允许使用链接表达式。				

键名称	说明	Type	必需	默认 值	可接受值
reboot	配置成功运行操作模块后的系统重启行为。 设置： • Force：在 msiexec 命令成功运行后启动系统重启。 • Allow：如果 msiexec 命令返回退出代码，表示需要重新启动，则启动系统重启。 • Skip：在 console.log 文件中记录一条信息性消息，表示跳过了重新启动。即使 msiexec 命令返回表示需要重新启动的退出代	字符串	否	Allow	Allow, Force, Skip

键名称	说明	Type	必需	默认 值	可接受值
	码，该选项也可防止重新启动。				
logOptions	指定用于 MSI 安装日志的选项。指定的标志与 /L 命令行参数一起传递给 MSI 安装程序，以启用日志记录。如果未指定任何标志，则 AWSTOE 使用默认值。 有关 MSI 日志选项的更多信息，请参阅 Microsoft Windows Installer 产品文档中的 命令行选项。	字符串	否	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

键名称	说明	Type	必需	默认 值	可接受值
logFile	日志文件位置的绝对路径或相对路径。如果该日志文件路径不存在，请创建它。如果未提供日志文件路径，则 AWSTOE 不存储 MSI 安装日志。	字符串	否	不适用	不适用

键名称	说明	Type	必需	默认 值	可接受值
properties	MSI 日志属性键值对，例如：TARGETDIR: "C:\target\location" 注意：不允许修改以下属性： • REBOOT="ReallySuppress" • REINSTALLMODE="ecmus" • REINSTALL="ALL"	Map[String]String	否	不适用	不适用

键名称	说明	Type	必需	默认 值	可接受值
ignoreAuthenticodeSignatureErrors	忽略路径中指定的安装程序的 authenticode 签名验证错误的标志。Get-AuthenticodeSignature 命令用于验证安装程序。 设置： • true：忽略验证错误并运行安装程序。 • false：不忽略验证错误。仅当验证成功时，安装程序才会运行。这是默认行为。	布尔值	否	false	true, false

键名称	说明	Type	必需	默认 值	可接受值
allowUnsignedInstaller	允许运行路径中指定的未签名安装程序的标志。Get-AuthenticodeSignature 命令用于验证安装程序。 设置： • true：忽略 Get-AuthenticodeSignature 命令返回的 NotSigned 状态并运行安装程序。 • false：需要对安装程序进行签名。未签名的安装程序将无法运行。这是默认行为。	布尔值	否	false	true, false

示例

以下示例显示了组件文档输入部分的变体，具体取决于您的安装路径。

输入示例：本地文档路径安装

```
- name: local-path-install
  steps:
    - name: LocalPathInstaller
      action: InstallMSI
      inputs:
        path: C:\sample.msi
        logFile: C:\msilogs\local-path-install.log
        logOptions: '*VX'
        reboot: Allow
        properties:
          COMPANYNAME: '"Amazon Web Services"'
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: true
```

输入示例：Amazon S3 路径安装

```
- name: s3-path-install
  steps:
    - name: S3PathInstaller
      action: InstallMSI
      inputs:
        path: s3://<bucket-name>/sample.msi
        logFile: s3-path-install.log
        reboot: Force
        ignoreAuthenticodeSignatureErrors: false
        allowUnsignedInstaller: true
```

输入示例：Web 路径安装

```
- name: web-path-install
  steps:
    - name: WebPathInstaller
      action: InstallMSI
      inputs:
        path: https://<some-path>/sample.msi
        logFile: web-path-install.log
        reboot: Skip
        ignoreAuthenticodeSignatureErrors: true
```

```
        allowUnsignedInstaller: false
```

Output

以下是 InstallMSI 操作模块的输出示例。

```
{
 "logFile": "web-path-install.log",
 "msiExitCode": 0,
 "stdout": ""
}
```

UninstallMSI (Windows)

UninstallMSI 操作模块允许您使用 MSI 文件删除 Windows 应用程序。您可以使用本地文件路径、S3 对象 URI 或 Web URL 指定 MSI 文件位置。重新启动选项可配置系统的重新启动行为。

AWSTOE 根据操作模块的输入参数生成msiexec命令。生成 msiexec 命令时，MSI 文件位置 (path) 和日志文件位置 (logFile) 将明确用双引号 (") 括起来。

以下 MSI 退出代码视为成功：

- 0（成功）
- 1605（ERROR_UNKNOWN_PRODUCT）
- 1614（ERROR_PRODUCT_UNINSTALLED）
- 1641（已启动重启）
- 3010（需要重启）

Input

键名称	说明	Type	必需	默认 值	可接受值
path	使用以下之一指定 MSI 文件位置： • 本地文件路径。路径	字符串	是	不适用	不适用

键名称	说明	Type	必需	默认 值	可接受值
	可以是绝对路径，也可以是相对路径。 • 有效的 S3 对象 URI。 • 符合 RFC 3986 标准的有效 HTTP/HTTPS 网址（建议使用 HTTPS）。 允许使用链接表达式。				

键名称	说明	Type	必需	默认 值	可接受值
reboot	配置成功运行操作模块后的系统重启行为。 设置： • Force：在 msiexec 命令成功运行后启动系统重启。 • Allow：如果 msiexec 命令返回退出代码，表示需要重新启动，则启动系统重启。 • Skip：在 console.log 文件中记录一条信息性消息，表示跳过了重新启动。即使 msiexec 命令返回表示需要重新启动的退出代	字符串	否	Allow	Allow, Force, Skip

键名称	说明	Type	必需	默认 值	可接受值
	码，该选项也可防止重新启动。				
logOptions	指定用于 MSI 安装日志的选项。指定的标志与 /L 命令行参数一起传递给 MSI 安装程序，以启用日志记录。如果未指定任何标志，则 AWSTOE 使用默认值。 有关 MSI 日志选项的更多信息，请参阅 Microsoft Windows Installer 产品文档中的 命令行选项。	字符串	否	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

键名称	说明	Type	必需	默认 值	可接受值
logFile	日志文件位置的绝对路径或相对路径。如果该日志文件路径不存在，请创建它。如果未提供日志文件路径，则 AWSTOE 不存储 MSI 安装日志。	字符串	否	不适用	不适用

键名称	说明	Type	必需	默认 值	可接受值
properties	MSI 日志属性键值对，例如：TARGETDIR: "C:\target\location" 注意：不允许修改以下属性： • REBOOT="ReallySuppress" • REINSTALLMODE="ecmus" • REINSTALL="ALL"	Map[String]String	否	不适用	不适用

键名称	说明	Type	必需	默认 值	可接受值
ignoreAuthenticodeSignatureErrors	忽略路径中指定的安装程序的 authenticode 签名验证错误的标志。Get-AuthenticodeSignature 命令用于验证安装程序。 设置： • true：忽略验证错误并运行安装程序。 • false：不忽略验证错误。仅当验证成功时，安装程序才会运行。这是默认行为。	布尔值	否	false	true, false

键名称	说明	Type	必需	默认 值	可接受值
allowUnsignedInstaller	允许运行路径中指定的未签名安装程序的标志。Get-AuthenticodeSignature 命令用于验证安装程序。 设置： • true：忽略 Get-AuthenticodeSignature 命令返回的 NotSigned 状态并运行安装程序。 • false：需要对安装程序进行签名。未签名的安装程序将无法运行。这是默认行为。	布尔值	否	false	true, false

示例

以下示例显示了组件文档输入部分的变体，具体取决于您的安装路径。

输入示例：移除本地文档路径安装

```
- name: local-path-uninstall
  steps:
    - name: LocalPathUninstaller
      action: UninstallMSI
      inputs:
        path: C:\sample.msi
        logFile: C:\msilogs\local-path-uninstall.log
        logOptions: '*VX'
        reboot: Allow
        properties:
          COMPANYNAME: '"Amazon Web Services"'
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: true
```

输入示例：移除 Amazon S3 路径安装

```
- name: s3-path-uninstall
  steps:
    - name: S3PathUninstaller
      action: UninstallMSI
      inputs:
        path: s3://<bucket-name>/sample.msi
        logFile: s3-path-uninstall.log
        reboot: Force
        ignoreAuthenticodeSignatureErrors: false
        allowUnsignedInstaller: true
```

输入示例：移除 Web 路径安装

```
- name: web-path-uninstall
  steps:
    - name: WebPathUninstaller
      action: UninstallMSI
      inputs:
        path: https://<some-path>/sample.msi
        logFile: web-path-uninstall.log
        reboot: Skip
        ignoreAuthenticodeSignatureErrors: true
```

```
        allowUnsignedInstaller: false
```

Output

以下是 UninstallMSI 操作模块的输出示例。

```
{
 "logFile": "web-path-uninstall.log",
 "msiExitCode": 0,
 "stdout": ""
}
```

系统操作模块

下一部分介绍了执行系统操作或更新系统设置的操作模块。

系统操作模块

- Reboot（Linux、Windows）
- SetRegistry（视窗）
- UpdateOS（Linux、Windows）

Reboot（Linux、Windows）

重启 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。默认情况下，delaySeconds 设置为 0，表示没有延迟。由于在实例重启时步骤超时不适用，因此重启操作模块不支持步骤超时。

如果该应用程序是由 Systems Manager 代理调用的，则向 Systems Manager 代理返回退出代码（Windows 为 3010，Linux 为 194）。Systems Manager 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： AWSTOE 创建了一个 Windows 任务调度器条目，其触发器在 SystemStartup 自动运行

- Linux： AWSTOE 在 crontab 中添加了一个在系统重启后自动运行的作业。

```
@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。

> **重试**
>
> 默认情况下，最大重试次数设置为 Systems Manager `CommandRetryLimit`。如果重启次数超过重试限制，自动化将失败。您可以通过编辑 Systems Manager 代理配置文件（`Mds.CommandRetryLimit`）来更改限制。请参阅 Systems Manager 代理开源中的 Runtime Configuration。

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。

Input

键名称	说明	Type	必需	默认
`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：重启步骤

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

输出

无。

在 重新引导 模块完成后，Image Builder 继续执行构建中的下一步。

SetRegistry（视窗）

SetRegistry操作模块接受输入列表，并允许您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。

Input

键名称	说明	Type	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。	String/Number/Array	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：设置注册表键值

```
  - 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。

UpdateOS（Linux、Windows）

UpdateOS 操作模块增加了对安装 Windows 和 Linux 更新的支持。默认情况下会安装所有可用更新。或者，您可以为待安装的操作模块配置一个或多个特定更新的列表。您也可以指定要从安装中排除的更新。

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

> **Note**
>
> UpdateOS 不支持亚马逊 Linux 2023 () AL2023。我们建议您将基本 AMI 更新到每个发布版本附带的新版本。有关其他替代方案，请参阅 Amazon Linux 2023 用户指南中的控制从主要版本和次要版本收到的更新。

- 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。你应当拥有运行 UpdateOS 操作模块的 `sudo` 权限。如果您没有 `sudo` 权限，则会返回 `error.Input`。

Input

键名称	说明	Type	必需
include	对于 Windows，可以指定以下值： • IDs要包含在可能安装的更新列表中的一篇或多篇 Microsoft 知识库 (KB) 文章。有效的格式为 KB1234567 或 1234567。 • 使用通配符值 (*) 的更新名称。有效的格式为 Security* 或 *Security* 。 对于 Linux，您可以指定一个或多个要包括在安装的更新列表中的软件包。	字符串列表	否
exclude	对于 Windows，可以指定以下值： • IDs要在安装中排除的更新列表中包含一篇或多篇 Microsoft 知识库 (KB) 文章。有效的	字符串列表	否

键名称	说明	Type	必需
	格式为 KB1234567 或 1234567。 • 使用通配符 (*) 值的更新名称。有效的格式为：Security* 或 *Security* 。 对于 Linux，您可以指定一个或多个要从安装的更新列表中排除的软件包。		

输入示例：添加对安装 Linux 更新的支持

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

输入示例：添加对安装 Windows 更新的支持

```
- name: UpdateWindowsOperatingSystem
  action: UpdateOS
  onFailure: Abort
  maxAttempts: 3
  inputs:
    include:
      - KB1234567
      - '*Security*'
```

输出

无。

配置 AWSTOE 运行命令的输入

要简化命令的命令行输入，可以在文件 AWSTOE run扩展名为 JSON 格式的输入配置文件中包含命令参数和选项的.json设置。 AWSTOE 可以从以下位置之一读取您的文件：

- 本地文件路径 (*./config.json*)。
- 一个 S3 存储桶 (*s3://<bucket-path>/<bucket-name>/config.json*)。

输入 run 命令时，您可以使用 --config 参数指定输入配置文件。例如：

```
awstoe run --config <file-path>/config.json
```

输入配置文件

输入配置 JSON 文件中包含所有设置的键值对，您可以通过 run 命令参数和选项直接提供这些设置。如果您在输入配置文件和 run 命令中都指定了一个设置作为参数或选项，则以下优先规则适用：

优先规则

1. 通过参数或选项直接提供给run命令的 AWS CLI设置会覆盖在输入配置文件中为相同设置定义的任何值。
2. 输入配置文件中的设置会覆盖组件的默认值。
3. 如果没有将其他设置传递到组件文档，则它可以应用默认值（如果存在）。

此规则有两个例外：文档和参数。这些设置在输入配置中和作为命令参数时的工作方式有所不同。如果使用输入配置文件，则不得直接向 run 命令指定这些参数。这样做会产生错误。

组件设置

输入配置文件包含以下设置。要简化文件，可以省略任何不需要的可选设置。除非另有说明，否则所有设置均为可选设置。

- cwIgnoreFailures（布尔值）-忽略日志中的 CloudWatch 日志失败。
- cwLogGroup（字符串）- CloudWatch 日志的LogGroup名称。
- cwLogRegion（字符串）-适用于 CloudWatch 日志的 AWS 区域。

- cwLogStream（字符串）- CloudWatch 日志的LogStream名称，用于指示将console.log文件流式传输到 AWSTOE 何处。
- d@@ ocumentS3 BucketOwner（字符串）— 基于 URI 的 S3 文档的存储桶所有者的账户 ID。
- 文档（对象数组，必填）— 一个 JSON 对象数组，代表 AWSTOE run命令正在运行的 YAML 组件文档。必须指定至少一个组件文档。

  对象由以下字段组成：

  - 路径（字符串，必填）-YAML 组件文档的文件位置。其必须是以下内容之一：
    - 本地文件路径 (*./component-doc-example.yaml*)。
    - 一个 S3; URI (s3*://bucket/key*)。
    - Image Builder 组件构建版本 ARN（arn: aws: imagebuilder: us-west--: component/ /2021.12.02/1）。*2:123456789012 my-example-component*
  - parameters（对象数组）-键值对对象的数组，每个代表 run 命令在运行组件文档时传入的特定于组件的参数。组件的参数是可选的。组件文档可能定义参数，也可能没有定义参数。

    对象由以下字段组成：

    - name（字符串，必填）-组件参数的名称。
    - value（字符串，必填）-要传递到组件文档的命名参数的值。

    要了解有关组件参数的更多信息，请参阅 在自定义组件文档中使用变量 页面的参数一节。
- executonId（字符串）-这是适用于执行当前 run 命令的唯一 ID。此 ID 包含在输出和日志文件名中，用于唯一标识这些文件，并将它们链接到当前的命令执行。如果省略此设置，则 AWSTOE 生成 GUID。
- LogDire ctory（字符串）- AWSTOE 存储此命令执行的所有日志文件的目标目录。默认情况下，该目录位于以下父目录中：TOE_<DATETIME>_<EXECUTIONID>。如果未指定日志目录，则 AWSTOE 使用当前工作目录 (.)。
- LogS3 BucketName（字符串）-如果组件日志存储在 Amazon S3 中（推荐），则将组件应用程序日志 AWSTOE 上传到此参数中命名的 S3 存储桶。
- LogS3 BucketOwner（字符串）-如果组件日志存储在 Amazon S3 中（推荐），则这是 AWSTOE 写入日志文件的存储桶的所有者账户 ID。
- LogS3 KeyPrefix（字符串）— 如果组件日志存储在 Amazon S3 中（推荐），则这是存储桶中日志位置的 S3 对象密钥前缀。
- parameters（对象数组）-键值对对象的数组，表示全局应用于当前 run 命令执行中包含的所有组件的参数。

  - name（字符串，必填）-全局参数的名称。
  - 值（字符串，必填）-要传递到所有组件文档的命名参数的值。
- phases（字符串）-以逗号分隔的列表，它指定要从 YAML 组件文档中运行哪些阶段。如果组件文档包含其他阶段，则这些阶段将无法运行。
- stateDirectory（字符串）-存储状态跟踪文件的文件路径。
- trace(布尔值)-启用对控制台的详细日志记录。

示例

以下示例显示了一个输入配置文件，该文件为两个组件文档 `sampledoc.yaml` 和 `conversation-intro.yaml` 运行 `build` 和 `test` 阶段。每个组件文档都有一个仅适用于其自身的参数，并且两者都使用一个共享参数。`project` 参数对两个组件文档均适用。

```
{
   "documents": [
     {
       "path": "<file path>/awstoe/sampledoc.yaml>",
       "parameters": [
         {
           "name": "dayofweek",
           "value": "Monday"
         }
       ]
     },
     {
       "path": "<file path>/awstoe/conversation-intro.yaml>",
       "parameters": [
         {
           "name": "greeting",
           "value": "Hello, HAL."
         }
       ]
     }
   ],
   "phases": "build,test",
   "parameters": [
     {
       "name": "project",
       "value": "examples"
     }
```

```
    ],
    "cwLogGroup": "<log_group_name>",
    "cwLogStream": "<log_stream_name>",
    "documentS3BucketOwner": "<owner_aws_account_number>",
    "executionId": "<id_number>",
    "logDirectory": "<local_directory_path>",
    "logS3BucketName": "<bucket_name_for_log_files>",
    "logS3KeyPrefix": "<key_prefix_for_log_files>",
    "logS3BucketOwner": "<owner_aws_account_number>"
}
```

Image Builder 输出映像资源

使用 Image Builder 为 AMI 或容器映像创建映像资源后，您可以使用 Image Builder 控制台、Image Builder API 或 AWS CLI中的 imagebuilder 命令对其进行管理。

> **Tip**
>
> 在您具有相同类型的许多资源时，制作标签可帮助您根据分配给特定资源的标签来识别它。有关使用 Image Builder 命令为资源添加标签的更多信息 AWS CLI，请参阅本指南的标记资源部分。

本部分介绍了如何列出、查看和创建映像。有关映像工作流以及如何对其进行管理的信息，请参阅 管理 Image Builder 映像的构建和测试工作流。

内容

列出映像和构建版本

在 Image Builder 控制台的映像页面上，您可以看到您拥有、与您共享以及您有权访问的所有 Image Builder 映像资源的列表。列表结果包括有关这些资源的一些关键详细信息。

您还可以查看账户中所有待处理工作流操作的映像。

内容

列出映像

本节介绍列出映像相关信息的不同方式。

您可以使用以下方法之一列出您有权访问的 Image Builder 映像资源。有关 API 操作，请参阅 I EC2 mage Builder API 参考ListImages中的。有关关联的 SDK 请求，请参阅同一页面上的另请参阅链接。

内容

在控制台中列出映像

要在控制台中打开映像列表页面，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择映像。

根据映像所有权或待处理的工作流操作，控制台中的映像页面分为多个选项卡。本节介绍前三个选项卡，这些选项卡显示了您拥有或有权访问的映像。

“控制台”选项卡：我拥有的

在我拥有的选项卡中，您可以使用以下筛选条件来简化映像列表结果。

- 您可以在搜索栏中搜索完整或部分名称。
- 您可以根据映像的操作系统平台（Windows、Linux 或 macOS）筛选映像。
- 您可以根据映像生成的输出类型（AMI 或容器映像）筛选映像。
- 您可以使用筛选源来查找从虚拟机 (VMIE) 或 ISO 磁盘映像导入的图像。

根据筛选条件控件，我拥有的选项卡会显示您创建的 Image Builder 映像列表，以及所列资源的以下详细信息：

名称/版本

Image Builder 映像资源名称以配方名称和其构建版本开头。选择该链接以查看所有相关的映像构建版本。

类型

Image Builder 在创建此映像资源（AMI 或容器映像）时的输出映像类型。

平台

映像资源的操作系统平台，例如“Linux”、“Windows”或“macOS”。

映像来源

Image Builder 用于构建此映像资源的基础映像的来源。主要用于筛选从虚拟机（VMIE）导入的映像结果。

Creation time

Image Builder 创建当前版本映像资源的日期和时间。

ARN

当前版本映像资源的 Amazon 资源名称（ARN）。

“控制台”选项卡：与我共享

在与我共享选项卡中，您可以使用以下筛选条件来简化映像列表结果。

- 您可以在搜索栏中搜索完整或部分名称。
- 您可以根据映像的操作系统平台（Windows、Linux 或 macOS）筛选映像。
- 您可以根据映像生成的输出类型（AMI 或容器映像）筛选映像。
- 您可以使用筛选源来查找从虚拟机 (VMIE) 或 ISO 磁盘映像导入的图像。

根据筛选条件控件，与我共享选项卡会显示与您共享的 Image Builder 映像列表，以及所列资源的以下详细信息：

映像名称

与您共享的映像资源的名称。要在配方中使用共享映像，请选择选择托管映像选项，然后将映像来源更改为与我共享的映像。

类型

Image Builder 在创建此映像资源（AMI 或容器映像）时的输出映像类型。

版本

映像资源的操作系统平台版本，通常是以下格式的数字字段：<major>.<minor>.<patch>。

映像来源

Image Builder 用于构建此映像资源的基础映像的来源（如适用）。主要用于筛选从虚拟机（VMIE）导入的映像结果。

平台

映像资源的操作系统平台，例如“Linux”、“Windows”或“macOS”。

Creation time

Image Builder 创建与您共享的映像资源版本的日期和时间。

所有者

共享映像资源的所有者。

ARN

与您共享的映像资源版本的 Amazon 资源名称（ARN）。

“控制台”选项卡：由 Amazon 管理

在由 Amazon 管理选项卡中，您可以使用以下筛选条件来简化映像列表结果。

- 您可以在搜索栏中搜索完整或部分名称。
- 您可以根据映像的操作系统平台（Windows、Linux 或 macOS）筛选映像。
- 您可以根据映像生成的输出类型（AMI 或容器映像）筛选映像。
- 您可以使用筛选源来查找从虚拟机 (VMIE) 或 ISO 磁盘映像导入的图像。

根据筛选条件控件，由 Amazon 管理选项卡会显示 Amazon 管理的 Image Builder 映像列表，您可以将其用作配方的基础映像。Image Builder 会显示所列资源的以下详细信息：

映像名称

托管映像的名称。创建配方时，基础映像的默认设置为快速启动（由 Amazon 管理）。此选项卡中列出的映像将填充与您在创建配方时为基础映像选择的操作系统平台关联的映像名称列表。

类型

Image Builder 在创建此映像资源（AMI 或容器映像）时的输出映像类型。

版本

映像资源的操作系统平台版本，通常是以下格式的数字字段：<major>.<minor>.<patch>。

平台

映像资源的操作系统平台，例如“Linux”、“Windows”或“macOS”。

Creation time

Image Builder 创建与您共享的映像资源版本的日期和时间。

所有者

Amazon 拥有这些托管映像。

ARN

与您共享的映像资源版本的 Amazon 资源名称（ARN）。

使用 AWS CLI 命令列出图像

在中运行list-images命令时 AWS CLI，您可以获得自己拥有或有权访问的图像的列表。

以下命令示例演示了如何使用不带筛选条件的 list-images 命令列出您拥有的所有 Image Builder 映像资源。

示例：列出所有映像

```
aws imagebuilder list-images
```

输出：

```
{
 "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
 "imageVersionList": [
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0",
    "name": "image-recipe-name",
```

```
    "type": "AMI",
    "version": "1.0.0",
    "platform": "Linux",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  },
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-win/1.0.1",
    "name": "image-recipe-win",
    "type": "AMI",
    "version": "1.0.1",
    "platform": "Windows",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  },
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
macos/1.1.1",
    "name": "image-recipe-macos",
    "type": "AMI",
    "version": "1.1.1",
    "platform": "macOS",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  }
 ]
}
```

运行 list-images 命令时，可以应用筛选器来简化结果，如以下示例所示。有关如何筛选结果的更多信息，请参阅AWS CLI 命令引用中的 list-images 命令。

示例：筛选 Linux 映像

```
aws imagebuilder list-images --filters name="platform",values="Linux"
```

输出：

```
{
 "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
 "imageVersionList": [
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0",
```

```
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0",
    "platform": "Linux",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  }
 ]
}
```

列出等待操作的映像

当您在映像工作流中使用 WaitForAction 步骤操作时，其会暂停工作流，直到您向其发送恢复处理或使工作流失败的信号。如果需要在继续操作前运行某个外部进程，则可以使用此步骤操作。然后，您可以使用 SendWorkflowStepAction 将暂停步骤的信号发送到 RESUME 或 STOP。您也可以通过控制台停止或恢复工作流。

以下选项卡显示了如何获取账户中所有映像资源的列表，其中包含当前暂停以等待信号恢复或停止的工作流步骤。这些选项卡涵盖了控制台步骤和 AWS CLI 命令。

您还可以使用 API 或 SDK 来获取等待操作的工作流步骤列表。有关 API 操作，请参阅 I EC2 mage Builder API 参考ListWaitingWorkflowSteps中的。有关关联的 SDK 请求，请参阅同一页面上的另请参阅链接。

Console

要进入控制台中的等待操作选项卡，请按照以下步骤操作：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择映像。此操作将打开映像列表页面。
3. 从列表页面中选择等待操作选项卡。
4. （可选）要停止或恢复步骤，请选中名称旁边的复选框，然后选择停止步骤或恢复步骤。您可以选中多个复选框，对所有选定步骤执行相同的操作。

待处理的工作流步骤详细信息

待处理步骤的工作流详细信息包括以下内容：

- 映像名称 – 包含待处理步骤的映像资源名称。您可以选择名称链接以显示该映像的详情页面。

- 待处理步骤名称 – 等待操作的工作流步骤名称。
- 步骤执行 ID – 对工作流步骤的运行时实例进行唯一标识。您可以选择链接的 ID 以显示该步骤的运行时详细信息。
- 步骤启动 – 工作流步骤的运行时实例启动时的时间戳。
- 工作流 ARN – 包含待处理步骤的工作流的 Amazon 资源名称（ARN）。
- 操作 – 处于等待状态的步骤操作。

AWS CLI

当你运行中的list-waiting-workflow-steps命令时 AWS CLI，你会得到一份清单，列出你账户中所有有工作流程步骤在完成图像创建过程之前等待操作的图片。

以下命令示例演示了如何使用 list-waiting-workflow-steps 命令列出您账户中的所有映像，其中包含正在等待操作的工作流步骤。

示例：列出账户中包含等待工作流步骤的映像

```
aws imagebuilder list-waiting-workflow-steps
```

输出：

此示例的输出显示了账户中的一个映像，其中有一个步骤正在等待操作。

```
{
    "steps": [
        {
            "imageBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:111122223333:image/example-image/1.0.0/8",
            "name": "WaitForAction",
            "workflowExecutionId": "wf-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "stepExecutionId": "step-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "workflowBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:111122223333:workflow/test/wait-for-action/1.0.0/1",
            "startTime": "2023-11-21T23:21:23.609Z",
            "action": "WaitForAction"
        }
    ]
}
```

列出映像构建版本

在 Image Builder 控制台的映像构建版本页面上，您可以看到构建版本列表以及您拥有的映像资源的更多详细信息。您还可以在 Image Builder API 中使用命令或操作 AWS CLI 来列出映像构建版本。SDKs

您可以使用以下方法之一列出您拥有的映像资源的映像构建版本。有关 API 操作，请参阅 I EC2 mage Builder API 参考ListImageBuildVersions中的。有关关联的 SDK 请求，请参阅同一页面上的另请参阅链接。

Console

版本详细信息

Image Builder 控制台中映像构建版本页面上的详细信息包括以下内容：

- 版本 - 映像资源构建版本。在 Image Builder 控制台中，该版本链接到映像详细信息页面。
- 类型 – Image Builder 在创建此映像资源（AMI 或容器映像）时分发的输出类型。
- 创建日期 – Image Builder 创建映像构建版本的日期和时间。
- 映像状态 - 映像构建版本的当前状态。状态可能与映像构建或处置有关。例如，在构建 过程中，您可能会看到`Building` 或 `Distributing` 的状态。有关映像的处置情况，您可能会看到 `Deprecated` 或的状态`Deleted`。
- 失败原因 – 映像状态的原因。Image Builder 控制台仅显示构建失败的原因（映像状态等于 `Failed`）。
- 安全调查结果 – 所引用映像构建版本的聚合映像扫描调查发现。
- ARN – 映像资源引用版本的 Amazon 资源名称（ARN）。
- 日志流 – 指向所引用映像构建版本的日志流详细信息的链接。

列出版本

要在 Image Builder 控制台中列出映像构建版本，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择 映像。默认情况下，映像列表显示您拥有的每个映像的当前版本。
3. 要查看映像的所有版本列表，请选择当前版本链接。该链接将打开映像构建版本页面，其中列出了特定映像的所有构建版本。

AWS CLI

在中运行list-image-build-versions命令时 AWS CLI，您将获得指定图像资源的完整构建版本列表。您必须是映像的拥有者才能执行此命令。

以下命令示例说明如何使用 list-image-build-versions 命令列出指定映像的所有构建版本。

示例：列出特定映像的构建版本

```
aws imagebuilder list-image-build-versions --image-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0
```

输出：

此示例的输出包括指定映像配方的两个构建版本。

```
{
  "requestId": "12f3e45d-67cb-8901-af23-45ed678c9b01",
  "imageSummaryList": [
    {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
name/1.0.0/2",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0/2",
    "platform": "Linux",
    "osVersion": "Amazon Linux 2",
    "state": {
      "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2023-03-10T01:04:40.609Z",
    "outputResources": {
      "amis": [
        {
        "region": "us-west-2",
        "image": "ami-012b3456789012c3d",
        "name": "image-recipe-name 2023-03-10T01-05-12.541Z",
        "description": "First verison of image-recipe-name",
        "accountId": "123456789012"
        }
      ]
    },
    "tags": {}
```

```
    },
    {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
name/1.0.0/1",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0/1",
    "platform": "Linux",
    "osVersion": "Amazon Linux 2",
    "state": {
      "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2023-03-10T00:07:16.384Z",
    "outputResources": {
      "amis": [
        {
        "region": "us-west-2",
        "image": "ami-0d1e23456789f0a12",
        "name": "image-recipe-name 2023-03-10T00-07-18.146132Z",
        "description": "First verison of image-recipe-name",
        "accountId": "123456789012"
        }
      ]
    },
    "tags": {}
    }
  ]
}
```

> **Note**
>
> list-image-build-versions 命令的输出目前不包括安全调查发现或日志流。

查看映像资源详细信息

在 Image Builder 控制台的映像详细信息页面上，您可以查看您拥有的特定映像资源的详细信息。您还可以在 Image Builder API 中使用命令或操作 AWS CLI 来获取图像详情。 SDKs

有关其他人通过 AWS Resource Access Manager (AWS RAM) 资源 AWS 账户 共享与您共享的资源的更多信息，请参阅AWS RAM 用户指南中的访问与您共享的 AWS 资源。

内容

在 Image Builder 控制台中查看映像详细信息

Image Builder 控制台中的映像详细信息页面包含摘要部分，其他信息按标签页分组。页面标题是创建映像的配方的名称和构建版本。如果某个选项卡不适用于您的图片，则该选项卡处于非活动状态且不显示数据。

控制台详细信息部分和标签页

- 摘要部分
- “输出资源”标签页
- 基础设施配置标签页
- 分配设置标签页
- “工作流程”标签页
- “安全调查发现”标签页
- “标签”选项卡

摘要部分

摘要部分跨越页面的宽度，包括以下详细信息。这些详细信息将始终显示。

配方

不包含构建版本的配方名称和版本。例如，如果构建版本是 `sample-linux-recipe | 1.0.1/2`，则配方是 `sample-linux-recipe | 1.0.1`，构建版本是 2。

Date created (创建日期)

Image Builder 创建映像构建版本的日期和时间。

映像状态

映像构建版本的当前状态。状态可能与映像构建或处置有关。例如，在构建 过程中，您可能会看到`Building` 或 `Distributing` 的状态。有关映像的处置情况，您可能会看到 `Deprecated` 或的状态`Deleted`。

失败的原因

映像状态的原因。Image Builder 控制台仅显示构建失败的原因（映像状态等于 `Failed`）。

“输出资源”标签页

输出资源标签页列出了当前显示的映像资源的输出和分配详细信息。Image Builder 显示的信息取决于管道用于创建映像的配方类型，如下所示。

映像配方

- 区域 — 在映像列中指定的亚马逊机器映像 (AMI) 的分配区域。
- 映像 — Image Builder 分配到目标的 AMI 的 ID。此 ID 链接到 Amazon EC2 控制台中的亚马逊系统映像 (AMIs) 页面。

> **Note**
>
> Image Builder 在创建输出映像资源之后以及将 AMI 分配到目标之前创建 AMI。

- 名称 — Image Builder 分配到目标的 AMI 的名称。
- 描述 — 管道用于创建输出映像资源的映像配方中的可选描述。
- 帐户-拥有当前显示 AWS 账户 的 Image Builder 图像资源的。

容器配方

Image Builder 显示从容器配方创建的输出的以下详细信息。

- 区域 — 在映像 URI 列中指定的容器映像的分配区域。
- 映像 URI — Image Builder 分配到目标区域的 ECR 存储库的输出容器映像的 URI。

> **Note**
>
> Image Builder 为每个目标显示一行。输出映像始终至少有一个条目可供分发给创建该映像的帐户。其他目的地可能包括跨区域的分布 AWS 账户、或 AWS Organizations。有关更多信息，请参阅 管理 Image Builder 分配设置。

基础设施配置标签页

基础设施配置标签页显示 Image Builder 用来构建和测试当前显示的映像的 Amazon EC2 基础设施设置。Image Builder 始终显示基础设施配置资源的名称（配置名称）及其 Amazon 资源名称 (ARN)。如果您的基础设施配置设置了值，则其他基础设施详细信息可能包括以下内容

- 实例类型
- 实例配置文件
- 网络基础设施
- 安全组设置
- Image Builder 存储应用程序日志的 Amazon S3 位置
- 用于故障排除的 Amazon EC2 密钥对
- 事件通知的 Amazon SNS 主题

有关更多信息，请参阅 管理 Image Builder 基础设施配置。

分配设置标签页

分配设置标签页显示 Image Builder 用于分配输出映像的设置。Image Builder 始终显示分配配置资源的名称（配置名称）及其 Amazon 资源名称 (ARN)。其他分配详细信息取决于 Image Builder 管道用于创建映像的配方类型，如下所示：

映像配方

如果您的分配配置资源设置了这些值，则其他分配详细信息可能包括以下内容：

- 区域 — 输出 Amazon Machine Image (AMI) 的分配区域。
- 输出 AMI 名称 — Image Builder 分配到目标的 AMI 的名称。
- 加密（KMS 密钥） — 如果已配置，Image Builder 将用来加密映像以分配到目标区域的 AWS KMS key 。
- 要@@ 分配的目标账户-如果您配置了跨账户分发，则此列将显示目标区域中 AWS 账户 要与之共享输出图像的逗号分隔列表。
- 拥有共享权限的委托人-以逗号分隔的有权启动您的映像的 AWS 委托人列表，例如、群组 AWS 账户 AWS Organizations 或组织单位 ()。OUs

> **Note**
>
> 当您授予其他委托人启动您的图像的权限时，您仍然拥有该映像。 AWS 向您的账户收取 Amazon EC2 从您的映像启动的所有实例的账单。

- 目标账户以加快启动配置 — EC2 Fast Launch AWS 账户 在其中分发预配置的快照以供启动。
- 关联的许可证配置 — 与指定区域中的 AMI 关联 ARNs 的 License Manager 许可证配置。
- 启动模板配置-标识用于特定账户的 Amazon EC2 启动模板。
- 设置启动模板默认版本-将指定的 Amazon EC2 启动模板设置为指定版本的默认启动模板 AWS 账户。

容器配方

容器分配始终包含以下详细信息：

- 区域 — 在映像 URI 列中指定的容器映像的分配区域。
- 映像 URI — Image Builder 分配到目标区域的 Amazon ECR 存储库的输出容器映像的 URI。

> **Note**
>
> Image Builder 为每个目标显示一行。输出映像始终至少有一个条目可供分发给创建该映像的帐户。其他目的地可能包括跨区域的分布 AWS 账户、或 AWS Organizations。有关更多信息，请参阅 管理 Image Builder 分配设置。

“工作流程”标签页

工作流程定义了 Image Builder 在创建新映像时执行的步骤顺序。所有映像都有构建、测试和分发工作流程。工作流程标签页显示 Image Builder 为映像运行的适用工作流程。

筛选工作流程类型

默认情况下，Image Builder 最初会显示构建或导入工作流程摘要和工作流程步骤。但是，工作流程筛选器会显示您的映像的所有正在进行或已完成的工作流程。要查看其他工作流程，请从列表中选择。

生成 AMI 输出的图像工作流程可以包含构建、导入或测试工作流程。生成容器输出的容器工作流程可以具有构建、测试或分发工作流程。

> **Note**
>
> 如果工作流程尚未启动，它将不会显示在列表中。例如，如果同时配置了构建和测试工作流程的映像构建刚刚开始，则生成工作流程是列表中唯一显示的工作流程类型。当测试工作流程开始时，Image Builder 会将其添加到列表中。

在工作流程筛选器之后，所选工作流程将显示运行时摘要，其中包含每种工作流程类型的以下详细信息：

工作流程状态

此工作流程的当前运行时状态。值可以包括：

- 待定
- Skipped
- 运行
- Completed
- 失败
- R ollback-in-progress
- 回滚已完成

执行 ID

Image Builder 分配的唯一标识符，用于在每次运行工作流程时跟踪运行时资源。

启动

工作流程的运行时实例启动的时间戳。

结束

工作流程的运行时实例完成的时间戳。

总步数

工作流程中的总步骤数。这应等于成功、跳过和失败的步骤的步数总和。

成功的步骤数

工作流程中成功运行的步骤数的运行时计数。

失败的步骤数

工作流程中失败步骤数的运行时计数。

跳过的步骤数

工作流程中跳过的步骤数的运行时计数。

以下列表中的详细信息报告了该工作流程的运行时实例中所有步骤的当前状态。Image Builder 为所有映像类型显示相同的详细信息。

步骤 #

一个编号，表示 Image Builder 运行工作流程步骤的顺序。

步骤 ID

工作流步骤的唯一标识符，在运行时分配。

步骤状态

指定工作流程步骤的当前运行时状态。

回滚状态

此工作流程的运行时实例失败时的当前回滚状态。

步骤名称

指定工作流程的名称。

启动

工作流程的运行时实例的指定步骤启动的时间戳。

结束

该工作流程的运行时实例的指定步骤完成的时间戳。

“安全调查发现”标签页

如果您已激活扫描，安全调查发现标签页会显示常见漏洞和风险敞口 (CVE) 调查结果。Amazon Inspector 在 Image Builder 为创建新映像而启动的测试实例上识别了这些调查发现。为确保 Image Builder 能够捕获映像的调查发现，必须按以下方式配置扫描：

1. 为您的账户激活 Amazon Inspector 扫描。有关更多信息，请参阅《Amazon Inspector 用户指南》中的 Amazon Inspector 入门。
2. 激活创建此映像的管道的安全调查发现。当您为管道激活安全调查发现时，Image Builder 会在终止测试实例之前保存调查发现的快照。有关更多信息，请参阅 在中为 Image Builder 图像配置安全扫描 AWS 管理控制台。

安全调查发现标签页包含 Amazon Inspector 为您的映像识别的每个漏洞的以下详细信息。

严重性

CVE 调查发现的严重性。值如下所示：

- 未分类
- 信息性
- 低
- 中
- 高
- 重大

调查发现 ID

Amazon Inspector 在扫描测试实例时为您的映像检测到的 CVE 调查发现的唯一标识符。该 ID 链接到安全调查发现 > 按漏洞页面。有关更多信息，请参阅 在中管理 Image Builder 图像的安全发现 AWS 管理控制台。

源

CVE 调查发现的漏洞信息来源。

天数

您的映像自首次观察到调查发现以来的天数。

Inspector 分数

Amazon Inspector 为 CVE 调查发现分配的分数。

“标签”选项卡

标签选项卡显示您为映像定义的所有标签。

从中获取图片政策详情 AWS CLI

以下示例说明了如何获取映像策略及其 Amazon 资源名称 (ARN) 的详细信息。

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

使用 Image Builder 创建自定义映像

您可以通过多种不同的方式创建新的 Image Builder 映像。例如，您可以使用以下方法之一使用 AWS 管理控制台 或创建图像 AWS CLI。您也可以使用 CreateImageAPI 操作或运行构建管道来创建映像。有关与 API 操作关联的 SDK 请求，可参阅《EC2 Image Builder API 参考》中该命令的 See Also 链接。

AWS 管理控制台

要从现有管道创建映像，您可以手动运行管道，如下所示。您也可以使用管道向导从头开始创建新映像。请参阅 管道向导：创建 AMI 或 管道向导：创建容器镜像，具体取决于您要创建的映像类型。

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在左侧导航窗格中，选择管道。
3. 选中要暂停的管道旁边的复选框。
4. 要创建映像，请从 “操作” 菜单中选择 “运行管道”。这会启动管道。

您还可以指定运行管道的时间表，或者根据您配置的规则使用 Amazon EventBridge 运行您的管道。

AWS CLI

在中运行create-image命令之前 AWS CLI，如果以下资源尚不存在，则必须创建这些资源：

所需的资源

- 配方-您必须为图片指定一个配方，如下所示：

  映像配方

  使用 `--image-recipe-arn` 参数为您映像配方资源指定 Amazon 资源名称（ARN）。

容器配方

使用 `--container-recipe-arn` 参数为您的容器配方资源指定 ARN。

- 基础设施配置-使用 `--infrastructure-configuration-arn` 参数为您的基础设施配置资源指定 ARN。

您还可以指定映像所需的以下任意资源：

可选资源和配置

- 分配配置-默认情况下，Image Builder 会将输出映像资源分配到您运行 create-image 命令的区域中的账户。要为您的分配提供其他目标或配置，请使用 `--distribution-configuration-arn` 参数为您的分配配置资源指定 ARN。
- 映像扫描 – 要为映像或容器测试实例上的 Amazon Inspector 调查发现配置快照，请使用 `--image-scanning-configuration` 参数。对于容器映像，您还可以指定 Amazon Inspector 作为其扫描用途的 ECR 存储库。
- 映像测试-要禁用 Image Builder 测试阶段，请使用 `--image-tests-configuration` 参数。或者，你可以为它可以运行的时间设置一个超时值。
- 映像标签 - 使用 `--tags` 参数向输出映像资源添加标签。
- 映像工作流 – 如果您未指定任何构建或测试工作流，则 Image Builder 将使用其默认映像工作流创建映像。要指定已创建的工作流，请使用 `--workflows` 参数。

> **Note**
>
> 如果指定映像工作流，则还必须在 `--execution-role` 参数中提供 Image Builder 用于运行工作流操作的 IAM 角色的名称或 ARN。

以下示例显示如何使用使用 create-image AWS CLI 命令创建映像。有关更多信息，请参阅 AWS CLI 命令参考。

示例：使用默认分配创建基本映像

```
aws imagebuilder create-image --image-recipe-arn arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/simple-recipe-linux/1.0.0 --infrastructure-configuration-arn arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/simple-infra-config-linux
```

输出：

```
{
"requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
"imageVersionList": [
    {
        "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/simple-recipe-
linux/1.0.0",
        "name": "simple-recipe-linux",
        ...
    }
]
}
```

从中取消图像创建 AWS CLI

要取消正在进行中的映像构建，请使用 cancel-image-creation 命令，如下所示：

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

使用 Image Builder 导入和导出虚拟机映像

当您从虚拟化环境中导出虚拟机时，该过程会创建一个或多个磁盘容器文件，这些文件充当虚拟机环境、设置和数据的快照。您可以使用这些文件导入 VM，并将其用作映像配方的基础映像。要导出，您可以创建虚拟机磁盘文件作为自定义映像版本的输出，然后分发这些文件。

对于 VM 磁盘容器，Image Builder 支持以下文件格式：

- 开放虚拟化归档 (OVA)
- 虚拟机磁盘 (VMDK)
- 虚拟硬盘 (VHD/VHDX)
- Raw

导入时使用磁盘来创建 Amazon Machine Image (AMI) 和 Image Builder 映像资源，这两者都可以用作自定义映像配方的基础映像。VM 磁盘必须存储在 S3 存储桶中才能导入。另外，也可以现有 EBS 快照中导入。

在 Image Builder 控制台中，您可以直接导入映像，然后在配方中使用输出映像或 AMI，也可以在创建配方或配方版本时指定导入参数。有关作为映像配方的一部分进行导入的更多信息，请参阅 虚拟机导入配置。

将 VM 导入 Image Builder

Image Builder 与 Amazon EC2 VM Import/Export API 集成，使导入过程能够在后台异步运行。Image Builder 会引用虚拟机导入中的任务 ID 来跟踪其进度，并创建一个 Image Builder 映像资源作为输出。这允许您在虚拟机导入完成之前在配方中引用 Image Builder 映像资源。

Console

要使用 Image Builder 控制台导入虚拟机，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择映像。
3. 要打开导入对话框，请选择导入图像。
4. 输入以下一般信息：
   - 为您的图片指定一个唯一的名称。
   - 指定基本映像的版本。采用以下格式：*major.minor.patch*。
5. 选择导入类型：虚拟机导入。
6. 在导入映像页面上提供以下每个部分的详细信息。完成操作后，选择导入映像。

基本映像操作系统

1. 选择与您的虚拟机操作系统平台匹配的映像操作系统 (OS) 选项。
2. 从列表中选择与您的虚拟机版本相匹配的操作系统版本。

虚拟机导入配置

1. 当您从虚拟化环境中导出虚拟机时，该过程会创建一个或多个磁盘容器文件。这些文件充当虚拟机环境、设置和数据的快照。您可以使用这些文件导入虚拟机作为映像配方的基本映像。有关 VMs在 Image Builder 中导入的更多信息，请参阅导入和导出 VM 映像。

   要指定导入源的位置，请执行以下步骤：

导入源

在磁盘容器 1 部分中指定要导入的第一个虚拟机映像磁盘容器或快照的来源。

a. 来源 — 可以是 S3 存储桶，也可以是 EBS 快照。

b. 选择磁盘的 S3 位置 – 输入 Amazon S3 中存储磁盘映像的位置。要浏览位置，请选择浏览 S3。

c. 要添加磁盘容器，请选择添加磁盘容器。

2. IAM 角色

要将 IAM 角色与您的虚拟机导入配置相关联，请从 IAM 角色下拉列表中选择该角色，或者选择创建新角色来创建一个新角色。如果您创建了新角色，IAM 角色控制台页面将在单独的标签页中打开。

3. 高级设置 – 可选

以下设置可选：使用这些设置，您可以为导入创建的基本映像配置加密、许可、标签等。

基本映像架构

要指定虚拟机导入源的架构，请从架构列表中选择一个值。

加密

如果您的虚拟机磁盘映像已加密，则必须提供用于导入过程的密钥。要为导入指定 KMS 密钥，请从加密（KMS 密钥）列表中选择一个值。该列表包含您的账户在当前区域中有权访问的 KMS 密钥。

许可证管理

导入虚拟机时，导入过程会自动检测虚拟机操作系统并将相应的许可证应用于基本映像。根据您的操作系统平台，许可证类型如下：

- 包含许可证 — 适用于您的平台的相应 AWS 许可证将应用于您的基本映像。
- 自带许可（BYOL） - 保留源自虚拟机的许可证（如果适用）。

要将使用创建的许可证配置附加 AWS License Manager 到您的基础映像，请从许可证配置名称列表中进行选择。有关 License Manager 的更多信息，请参阅使用 AWS License Manager

> **Note**
>
> - 许可证配置包含基于您的企业协议条款的许可规则。
> - Linux 仅支持 BYOL 许可证。

标签（基本映像）

标签使用键值对为您的 Image Builder 资源分配可搜索的文本。要为导入的基本映像指定标签，请使用键和值框输入键值对。

要添加标签，请选择 Add tag（添加标签）。要删除标签，请选择 Remove tag（删除标签）。

AWS CLI

要将虚拟机从磁盘导入 AMI 并创建可以立即引用的 Image Builder 映像资源，请在 AWS CLI中执行以下操作：

1. 使用中的 Amazon EC2 虚拟机 Import/Export import-image命令启动虚拟机导入 AWS CLI。记下命令响应中返回的任务 ID。下一步中您将需要使用该值。有关更多信息，请参阅《虚拟机 Import/Export 用户指南》中的 “使用虚拟机导入/导出” 将虚拟机作为映像导入。

2. 创建 CLI 输入 JSON 文件

   为了简化中使用的 Image Builder import-vm-image 命令 AWS CLI，我们创建了一个 JSON 文件，其中包含我们要传递到命令中的所有导入配置。

   > **Note**
   >
   > JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 操作请求参数，请参阅 I EC2 mage Builder API 参考中的ImportVmImage操作。

要将数据值作为命令行参数提供，请参阅AWS CLI 命令参考中指定的参数名。将 Image Builder import-vm-image 命令作为选项。

以下是我们在此示例中指定的参数的摘要：

- 名称（字符串，必填）— 要从导入中作为输出而创建的 Image Builder 映像资源的名称。
- semanticVersion（字符串，必填）— 输出映像的语义版本，按以下格式指定版本，每个位置都有数值表示特定版本：<major>.<minor>.<patch>。例如 `1.0.0`。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅 Image Builder 中的语义版本控制。
- 描述（字符串）— 映像配方的描述。
- 平台（字符串，必填）— 导入的 VM 的操作系统平台。
- vmImportTaskID（字符串，必填）— Amazon EC2 虚拟机导入过程中的 `ImportTaskId` (AWS CLI)。Image Builder 会监控导入过程，以提取其创建的 AMI，并构建可立即用于配方的 Image Builder 映像资源。
- 标签（字符串映射）— 标签是附加到导入资源的键值对。最多允许 50 个键值对。

将文件另存为 `import-vm-image.json`，以便在 Image Builder import-vm-image 命令中使用。

```
{
    "name": "example-request",
    "semanticVersion": "1.0.0",
    "description": "vm-import-test",
    "platform": "Linux",
    "vmImportTaskId": "import-ami-01ab234567890cd1e",
    "tags": {
     "Usage": "VMIE"
    }
}
```

3. 导入映像

使用您创建的文件作为输入并运行 import-vm-image 命令：

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

从您的映像版本中分发虚拟机磁盘 AWS CLI

在常规映像构建过程中，您可以使用 AWS CLI中的 Image Builder 分配配置，设置将支持的 VM 磁盘格式文件分配到目标区域的 S3 存储桶。有关更多信息，请参阅 示例：为来自的输出 VM 磁盘创建分发设置 AWS CLI。

使用 Image Builder 导入经过验证的 Windows ISO 磁盘映像

Windows 操作系统 ISO 文件是一个磁盘映像文件，其中包含特定版本的 Windows 操作系统的完整安装包。微软提供官方的Windows操作系统ISO文件供下载，可以直接从其网站下载，也可以通过授权经销商下载。请务必确保从可信的合法来源获取 ISO 文件，以避免潜在的恶意软件或未经授权的版本。

EC2 Image Builder 使用导入工作流程导入 ISO 磁盘文件并从中创建辅助卷。`build-image-from-iso`配置完成后，Image Builder 会拍摄其在导入时创建的卷的快照，并使用它来创建亚马逊系统映像 (AMI)。

ISO 磁盘映像导入支持的操作系统

Image Builder 支持以下 Windows 操作系统 ISO 磁盘映像：

- Windows 11 企业版 25H2 (x64)
- Windows 11 企业版 24H2 (x64)
- Windows 11 企业版 23H2 (x64)

Image Builder 不支持以下 Windows 操作系统 ISO 磁盘映像：

- 长期服务渠道 (LTSC) 映像
- 通过 Windows 媒体创建工具创建的 ISO 磁盘映像

- 评估图片

导入 ISO 磁盘映像的先决条件

要导入 ISO 磁盘映像，必须首先满足以下先决条件：

- 磁盘映像的操作系统必须是 Image Builder 支持的操作系统。有关支持的操作系统的列表，请参阅ISO 磁盘映像导入支持的操作系统。
- 为确保您可以导入 ISO 映像，请从微软 365 管理中心下载该映像。
- 在运行导入过程之前，您必须将 ISO 磁盘文件上传到 Amazon S3，该文件 AWS 区域 所在位置 AWS 账户 和导入运行位置。
- 文件扩展名在导入过程中区分大小写，并且必须区分大小写.ISO。如果您的文件扩展名为小写，则可以运行以下命令之一对其进行重命名：

  Command

  ```
  aws s3 cp s3://amzn-s3-demo-bucket/Win11_24H2_English.iso s3://amzn-s3-demo-bucket/Win11_24H2_English.ISO
  ```

  PowerShell

  ```
  Copy-S3Object -BucketName amzn-s3-demo-bucket -Key Win11_24H2_English.iso -DestinationKey Win11_24H2_English.ISO
  ```

- Microsoft 许可不会自动包含在导入中。您必须自带许可证 (BYOL)。有关微软软件许可的更多信息，请参阅亚马逊 Web Services 和 Microsoft 常见问题解答页面上的许可。
- 导入过程使用两个单独的 IAM 角色，如下所示：

  执行角色

  此角色授予 Image Builder AWS 服务 代表您调用的权限。您可以指定AWSServiceRoleForImageBuilder服务相关角色，其中包括执行角色所需的权限，也可以创建自己的角色。

  实例配置文件角色

  此角色授予服务在 EC2 实例上执行的操作的权限。您可以在基础设施配置资源中指定实例配置文件角色。将以下托管策略附加到您的实例配置文件角色，以确保您拥有导入过程所需的所有权限。

  - EC2InstanceProfileForImageBuilder

- AmazonSSMManagedInstanceCore

有关更多信息，请参阅 管理 Image Builder 基础设施配置。

将 ISO 磁盘映像导入 Image Builder

在开始导入过程之前，请确保您已满足所有要求先决条件。

导入过程会在您的映像上安装以下软件和驱动程序：

- EC2启动 v2
- AWS Systems Manager 代理人
- AWS NVMe 司机
- AWS ENA 网络驱动程序
- AWS PCI 串行驱动程序
- EC2 Windows 实用程序驱动程序
- 微软 Defender 更新套件

导入过程会对您的映像进行以下配置更新：

- 将系统配置为使用 Amazon Time 服务器。

Console

要使用 Image Builder 控制台导入 ISO 磁盘映像，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择映像。
3. 要打开导入对话框，请选择导入图像。
4. 输入以下一般信息：
   - 为您的图片指定一个唯一的名称。
   - 指定基本映像的版本。采用以下格式：*major.minor.patch*。
5. 选择导入类型：ISO 导入。
6. 输入以下 ISO 导入配置详细信息。完成操作后，选择导入映像。

- S3 URI-输入存储 ISO 磁盘文件的位置。要浏览文件，请选择 “浏览 S3”。
- IAM 角色-要将 IAM 角色与您的导入配置关联，请从 IAM 角色下拉列表中选择该角色，或者选择创建新角色来创建新角色。如果您创建了新角色，IAM 角色控制台页面将在单独的标签页中打开。

  您可以指定AWSServiceRoleForImageBuilder服务相关角色，也可以为服务访问指定自己的自定义角色。

7. 您可以选择向 Image Builder 图像资源添加标签。这不会将标签添加到您的 AMI。
8. ISO 基础架构配置定义了 Image Builder 为托管导入过程而启动的实例的设置。您可以使用 Image Builder 根据服务默认值创建的基础架构配置，也可以使用现有基础设施配置。有关更多信息，请参阅 管理 Image Builder 基础设施配置。

   要创建新的基础架构配置，请选择创建基础架构配置。这将在单独的选项卡中打开。创建完新资源后，可以返回到导入配置，然后选择 “使用现有基础架构配置”。
9. 要开始导入过程，请选择导入图像。

导入完成后，您的图像将出现在您拥有的图像列表中。有关更多详细信息，请参阅 列出映像。

AWS CLI

此示例说明如何从 ISO 磁盘文件导入映像并使用该映像创建 AMI AWS CLI。

以下是我们在此示例中指定的参数的摘要：

- 名称（字符串，必填）— 要从导入中作为输出而创建的 Image Builder 映像资源的名称。
- semanticVersion（字符串，必填）— 输出映像的语义版本，按以下格式指定版本，每个位置都有数值表示特定版本：<major>.<minor>.<patch>。例如 `1.0.0`。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅 Image Builder 中的语义版本控制。
- 描述（字符串）— 映像配方的描述。
- e@@ xecutionRole（字符串）— IAM 角色的名称或亚马逊资源名称 (ARN)，该角色授予 Image Builder 访问权限，以执行从微软 ISO 文件导入图像的工作流程操作。您可以指定AWSServiceRoleForImageBuilder服务相关角色，也可以为服务访问指定自己的自定义角色。
- 平台（字符串，必填）-ISO 磁盘映像的操作系统平台。有效值包括 `Windows`。
- osVersion（字符串，必填）-ISO 磁盘映像的操作系统版本。有效值包括 `Microsoft Windows 11`。

- infrastructureConfigurationArn（字符串，必填）— 用于启动构建 ISO 映像的EC2 实例的基础设施配置资源的 Amazon 资源名称 (ARN)。
- uri（字符串，必填）— 存储在 Amazon S3 中的 ISO 磁盘文件的 URI。

```
aws imagebuilder import-disk-image \
    --name "example-iso-disk-import" \
    --semantic-version "1.0.0" \
    --description "Import an ISO disk image" \
    --execution-role "AWSServiceRoleForImageBuilder" \
    --platform "Windows" \
    --os-version "Microsoft Windows 11" \
    --infrastructure-configuration-arn "arn:aws:imagebuilder:us-
east-1:111122223333:infrastructure-configuration/example-infrastructure-
configuration-123456789abc",
    --uri: "s3://amzn-s3-demo-source-bucket/examplefile.iso"
```

导入完成后，您的图像将出现在您拥有的图像列表中。有关更多详细信息，请参阅 列出映像。

PowerShell

此示例说明如何从 ISO 磁盘文件导入映像并使用该映像创建 AMI PowerShell。

以下是我们在此示例中指定的参数的摘要：

- 名称（字符串，必填）— 要从导入中作为输出而创建的 Image Builder 映像资源的名称。
- semanticVersion（字符串，必填）— 输出映像的语义版本，按以下格式指定版本，每个位置都有数值表示特定版本：<major>.<minor>.<patch>。例如 `1.0.0`。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅 Image Builder 中的语义版本控制。
- 描述（字符串）— 映像配方的描述。
- e@@ xecutionRole（字符串）— IAM 角色的名称或亚马逊资源名称 (ARN)，该角色授予 Image Builder 访问权限，以执行从微软 ISO 文件导入图像的工作流程操作。您可以指定AWSServiceRoleForImageBuilder服务相关角色，也可以为服务访问指定自己的自定义角色。
- 平台（字符串，必填）-ISO 磁盘映像的操作系统平台。有效值包括 `Windows`。
- osVersion（字符串，必填）-ISO 磁盘映像的操作系统版本。有效值包括 `Microsoft Windows 11`。
- infrastructureConfigurationArn（字符串，必填）— 用于启动构建 ISO 映像的EC2 实例的基础设施配置资源的 Amazon 资源名称 (ARN)。
- uri（字符串，必填）— 存储在 Amazon S3 中的 ISO 磁盘文件的 URI。

```
Import-EC2IBDiskImage `
    -Name "example-iso-disk-import" `
    -SemanticVersion "1.0.0" `
    -Description "Import an ISO disk image" `
    -ExecutionRole "AWSServiceRoleForImageBuilder" `
    -Platform "Windows" `
    -OsVersion "Microsoft Windows 11" `
    -InfrastructureConfigurationArn "arn:aws:imagebuilder:us-
east-1:111122223333:infrastructure-configuration/example-infrastructure-
configuration-123456789abc" `
    -Uri "s3://amzn-s3-demo-source-bucket/examplefile.ISO"
```

导入完成后，您的图像将出现在您拥有的图像列表中。有关更多详细信息，请参阅 列出映像。

从输出 AMI 启动实例

当你从导入过程创建的 AMI 启动实例时，Windows 操作系统会运行 Sysprep Specialize，它会自动从公共 S3 端点下载和安装 La EC2 unch v2 和 Systems Manager 代理。这些端点需要公共互联网接入。如果您从私有子网启动实例，则 Sysprep Specialize 进程将无法访问 S3 终端节点，并且启动失败。

管理 Image Builder 映像的安全调查发现

当您使用 Amazon Inspector 激活安全扫描时，它会持续扫描您账户中的计算机映像和正在运行的实例，以查找操作系统和编程语言的漏洞。如果激活，则安全扫描将自动进行，Image Builder 可以在您创建新映像时保存测试实例中的调查发现快照。Amazon Inspector 是一项付费服务。

当 Amazon Inspector 发现您的软件或网络设置中存在的漏洞时，它会采取以下措施：

- 通知您有调查发现。
- 评估结果的严重性。严重性评级对漏洞进行分类，以帮助您确定调查发现的优先级，并包括以下值：
  - 未分类
  - 信息性
  - 低
  - 中
  - 高
  - 重大

- 提供有关调查发现的信息，并提供指向其他资源的链接以获取更多详细信息。
- 提供补救指导，帮助您解决导致调查发现的问题。

在中为 Image Builder 图像配置安全扫描 AWS 管理控制台

如果您已为自己的账户激活了 Amazon Inspector，Amazon Inspector 会自动扫描 Image Builder 启动的 EC2 实例以构建和测试新映像。在构建和测试过程中，这些实例的生命周期很短，它们的调查发现通常会在这些实例关闭后立即过期。为了帮助您调查和修复新映像的调查发现，Image Builder 可以选择将 Amazon Inspector 在构建过程中在测试实例上识别的任何调查发现保存为快照。

第 1 步：为您的账户激活 Amazon Inspector 安全扫描

要从 Image Builder 控制台为您的账户激活 Amazon Inspector 安全扫描，请按照以下步骤操作：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择安全扫描设置。这将打开安全扫描对话框。

   该对话框显示您账户的扫描状态。如果已为您的账户激活 Amazon Inspector，则状态将显示为已启用。
3. 按照说明中的步骤 1 和步骤 2 激活 Amazon Inspector 扫描。

> **Note**
>
> Amazon Inspector 会产生费用。有关更多信息，请参阅 Amazon Inspector 定价。

如果您已激活对管道的扫描，Image Builder 会在您创建新映像时为您的构建实例拍摄调查发现的快照。这样，您就可以在 Image Builder 终止构建实例后访问调查发现。

第 2 步：将管道配置为保存漏洞调查发现的快照

要为管道配置漏洞调查发现快照，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择映像管道。
3. 选择以下方法之一以指定管道详细信息：

创建新管道

1. 在映像管道页面中，选择创建映像管道。这将在管道向导中打开指定管道详细信息页面。

更新现有管道

1. 在映像管道页面中，选择要更新的管道的管道名称链接。这将打开管道的详细页面。

> **Note**
>
> 或者，也可以选中要更新的管道之名称旁的复选框，然后选择查看详细信息。

2. 在管道详细信息页面上，从操作菜单中选择编辑管道。这会使您转至编辑管道页面。

4. 在管道向导的常规部分或 编辑管道页面中，选中启用安全扫描复选框。

> **Note**
>
> 如果您想稍后关闭快照，则可以编辑管道以清除该复选框。这不会停用 Amazon Inspector 对您账户的扫描。要停用 Amazon Inspector 扫描，请参阅 Amazon Inspector 用户指南中的t停用 Amazon Inspector。

在中管理 Image Builder 图像的安全发现 AWS 管理控制台

安全调查发现列表页面显示有关您的资源调查发现的高级信息，其视图基于您可以应用的几个不同的筛选条件。每个视图的顶部都包含以下用于更改视图的选项：

- 所有安全调查发现 – 如果您从 Image Builder 控制台的导航窗格中选择安全调查发现页面，则这是默认视图。
- 按漏洞 — 此视图显示您账户中所有有调查发现的映像资源的高级列表。调查发现 ID 链接到有关该调查发现的更多详细信息。此信息显示在页面右侧打开的面板上。面板包含以下信息：
  - 调查发现的详细说明。
  - 调查发现详情标签页。此标签页包括调查发现概述、受影响的软件包、补救建议摘要、漏洞详细信息和相关漏洞。漏洞 ID 链接到国家漏洞数据库中的详细漏洞信息。
  - 分数明细标签页。此选项卡包含 CVSS 和 Amazon Inspector 分数的side-by-side 比较，以便您可以查看 Amazon Inspector 在哪里修改了分数（如果适用）。

- 按映像管道 — 此视图显示您账户中每个映像渠道的调查发现数。Image Builder 显示中等严重性和更高程度的调查发现的数量数，以及所有调查发现的总数。列表中的所有数据都已链接，如下所示：
  - 映像管道名称列链接到指定映像管道的详情页面。
  - 严重性级别列链接打开所有安全调查发现视图，该视图按关联的映像管道名称和严重性级别进行筛选。

  您还可以使用搜索条件来优化结果。
- 按映像 — 此视图显示您账户中每个映像构建的调查发现数。Image Builder 显示中等严重性和更高程度的调查发现的数量数，以及所有调查发现的总数。列表中的所有数据都已链接，如下所示：
  - 映像名称列链接到指定映像构建的映像详情页面。有关更多信息，请参阅 查看映像资源详细信息。
  - 严重性级别列链接打开所有安全调查发现视图，该视图按关联的映像构建名称和严重性级别进行筛选。

  您还可以使用搜索条件来优化结果。

Image Builder 在默认所有安全调查发现视图的调查发现列表部分中显示以下详细信息。

严重性

CVE 调查发现的严重性。值如下所示：

- 未分类
- 信息性
- 低
- 中
- 高
- 重大

调查发现 ID

Amazon Inspector 在扫描构建实例时为您的映像检测到的 CVE 调查发现的唯一标识符。该 ID 链接到安全调查发现 > 按漏洞页面。

映像 ARN

在调查发现 ID 列中指定了调查发现的映像的 Amazon 资源名称（ARN）。

管道

构建映像 ARN 列中指定的映像的管道。

描述

调查发现的简短描述。

Inspector 分数

Amazon Inspector 为 CVE 调查发现分配的分数。

修复

链接到有关修复调查发现的建议行动方针的详细信息。

发布日期

此漏洞首次添加到供应商数据库的日期和时间。

清理 Image Builder 资源

为避免意外费用，请务必清理根据本指南中的示例创建的资源和管道。有关删除 Image Builder 中的资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。

管理 Image Builder 映像的生命周期策略

创建自定义映像时，重要的是要制定计划，在这些映像过时之前将其停用。Image Builder 管道可自动应用更新和安全补丁。但是，每次构建都会创建映像的新版本及其分配的所有关联资源。早期版本将保留在您的账户中，直到您手动将其删除，或创建脚本来执行任务。

借助 Image Builder 生命周期管理策略，您可以自动执行弃用、禁用和删除过时的映像及其关联资源的过程。关联的资源可以包括您分发给其他人 AWS 账户、组织和组织单位 (OUs) 的输出图像 AWS 区域。您可以定义规则，规定如何以及何时执行生命周期过程中的每个步骤，以及要在策略中包含哪些步骤。

自动化生命周期管理的优势

自动化生命周期管理的总体优势包括：

- 通过自动停用映像和关联资源的方式，简化自定义映像的生命周期管理。
- 有助于防止使用过时映像启动新实例所带来的合规风险。
- 删除过时映像来保持映像库存的新鲜度。
- 可以选择性删除已删除映像的关联资源，从而降低存储和数据传输成本。

实现成本节省

使用 EC2 Image Builder 创建自定义 AMI 或容器映像，无需任何费用。但是，标准定价仍适用于该过程中使用的其他服务。当您从中移除未使用或过时的图像及其相关资源时 AWS 账户，可以通过以下方式节省时间和成本：

- 不同时修补未使用或过时的映像时，可以减少修补现有映像所需的时间。
- 对于您删除的 AMI 图像资源，您可以选择同时移除分布式快照AMIs 及其关联的快照。这种方法可以节省存储快照的成本。
- 对于您删除的容器映像资源，可以选择删除底层资源。这种方法可以节省存储在 ECR 存储库中 Docker 映像的 Amazon ECR 存储成本和数据传输费率。

> **Note**
>
> Image Builder 无法评估所有可能的下游依赖项（例如自动扩缩组或启动模板）的潜在影响。配置策略操作时，必须考虑映像的下游依赖项。

内容

Image Builder 映像的生命周期管理先决条件

您必须满足以下先决条件，才能为映像资源定义 EC2 Image Builder 生命周期管理策略和规则。

- 创建一个 IAM 角色，该角色可授予 Image Builder 运行生命周期策略的权限。您可以通过以下方式之一创建此角色：
  - 创建生命周期策略时，使用 Image Builder 控制台中的 “使用服务默认值创建生命周期执行角色” 选项。这会自动创建一个附加`EC2ImageBuilderLifecycleExecutionPolicy`托管策略的角色。
  - 使用 Image Builder 控制台中的 “创建新的生命周期执行角色” 选项，这将打开 IAM，其中包含预先填写的设置，便于一键创建角色。
  - 在 IAM 控制台中手动创建角色。有关 step-by-step说明，请参阅为 Image Builder 生命周期管理创建 IAM 角色。
- 在目标账户中为跨账户分配的关联资源创建 IAM 角色。该角色可授予 Image Builder 在目标账户中对关联资源执行生命周期操作的权限。要创建该角色，请参阅为 Image Builder 跨账户生命周期管理创建 IAM 角色。

  > **Note**
  >
  > 如果您已为输出 AMI 授予启动权限，则不适用此先决条件。借助启动权限，您与之共享的账户拥有从共享 AMI 启动的实例，但所有 AMI 资源仍保留在您的账户中。

- 对于容器映像，您必须将以下标签添加到 ECR 存储库，向 Image Builder 授予访问权限，以便对存储在存储库中的容器映像运行生命周期操作：`LifecycleExecutionAccess: EC2 Image Builder`。

为 Image Builder 生命周期管理创建 IAM 角色

要授予 Image Builder 运行生命周期策略的权限，您必须先创建用于执行生命周期操作的 IAM 角色。按照以下步骤创建授予权限的服务角色。

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 从导航窗格中，选择 Roles。
3. 选择创建角色。此操作将进入流程的第一步选择可信实体，以创建您的角色。
4. 对于可信实体类型，请选择自定义信任策略选项。
5. 复制以下 JSON 信任策略，将其粘贴到自定义信任策略文本区域中，从而替换示例文本。此信任策略允许 Image Builder 担任您为运行生命周期操作而创建的角色。

   JSON

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

6. 从列表中选择以下托管策略：EC2ImageBuilderLifecycleExecutionPolicy，然后选择下一步。此操作将打开命名、检查并创建页面。

> **Tip**
>
> 筛选 `image` 以简化结果。

7. 输入角色名称。

8. 查看设置后，请选择创建角色。

为 Image Builder 跨账户生命周期管理创建 IAM 角色

要授予 Image Builder 在目标账户中对关联资源执行生命周期操作的权限，您必须首先创建其用于在这些账户中执行生命周期操作的 IAM 角色。您必须在目标账户中创建角色。

按照以下步骤创建在目标账户中授予权限的服务角色。

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 从导航窗格中，选择 Roles。
3. 选择创建角色。此操作将进入流程的第一步选择可信实体，以创建您的角色。
4. 对于可信实体类型，请选择自定义信任策略选项。
5. 复制以下 JSON 信任策略，将其粘贴到自定义信任策略文本区域中，从而替换示例文本。此信任策略允许 Image Builder 担任您为运行生命周期操作而创建的角色。

> **Note**
>
> 当 Image Builder 在目标账户中使用该角色对跨账户分配的关联资源进行操作时，其代表目标账户所有者行事。您在信任策略`aws:SourceAccount`中配置的账户是 Image Builder 分发这些资源的账户。 AWS 账户

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "imagebuilder.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "444455556666"
                },
                "StringLike": {
```

```
                    "aws:SourceArn": "arn:*:imagebuilder:*:*:image/*/*/*"
                }
            }
        }
    ]
}
```

6. 从列表中选择以下托管策略：EC2ImageBuilderLifecycleExecutionPolicy，然后选择下一步。此操作将打开命名、检查并创建页面。

> **Tip**
>
> 筛选 image 以简化结果。

7. 输入 Ec2ImageBuilderCrossAccountLifecycleAccess 作为角色名称。

> **Important**
>
> Ec2ImageBuilderCrossAccountLifecycleAccess 必须是该角色的名称。

8. 查看设置后，请选择创建角色。

列出 Image Builder 映像资源的生命周期管理策略

您可以获取图像生命周期管理策略列表，其中包括生命周期策略列表页面上的关键详细信息列 AWS 管理控制台，或者在 Image Builder API 中获取命令或操作 SDKs、或 AWS CLI。

您可以使用以下方法之一列出您 AWS 账户中的 Image Builder 映像生命周期策略资源。有关 API 操作，请参阅 EC2 Image Builder API 参考ListLifecyclePolicies中的。有关关联的 SDK 请求，请参阅同一页面上的另请参阅链接。

AWS 管理控制台

控制台中显示了现有策略的以下详细信息。您可以选择任何列来更改结果的排序顺序。策略列表最初按策略名称进行排序。当前排序顺序的列名以粗体显示。

如果结果不止一页，则面板右上角的分页箭头将变为活动状态。您可以通过搜索栏按策略名称、策略状态、输出映像类型和映像资源 ARN 筛选结果。

- 策略名称 – 策略的名称。

- 策略状态 – 策略处于活动状态还是非活动状态。
- 类型 – 创建新映像版本（AMI 或容器映像）时 Image Builder 分配的输出映像类型。
- 上次执行日期 – 生命周期策略上次运行的时间。
- 创建日期 – 生命周期策略创建以来的时间戳。
- ARN – 生命周期策略资源的 Amazon 资源名称（ARN）。

要在中列出生命周期策略 AWS 管理控制台，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择生命周期策略。这将显示您账户中的映像生命周期策略列表。

可用操作

您也可以从生命周期策略列表页面对生命周期策略执行以下操作。

要创建新的映像生命周期策略，请选择创建生命周期策略。有关如何创建策略的更多信息，请参阅创建生命周期策略。

要执行以下所有操作，必须首先选择策略。要选择策略，可以选中策略名称旁的复选框。

- 要关闭或打开策略，请从操作菜单中选择禁用策略或启用策略。
- 要更改策略，请从操作菜单中选择编辑策略。
- 要删除策略，请从操作菜单中选择删除策略。
- 要创建使用所选策略作为基准设置的新策略，请从操作菜单中选择克隆策略。

AWS CLI

以下命令示例说明如何使用列 AWS CLI 出特定镜像生命周期策略 AWS 区域。有关可用于此命令的参数和选项的更多信息，请参阅《list-lifecycle-policies命令参考》中的 AWS CLI 命令。

示例：

```
aws imagebuilder list-lifecycle-policies \
--region us-west-1
```

输出：

```
{
    "lifecyclePolicySummaryList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy1",
            "name": "sample-lifecycle-policy1",
            "status": "DISABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-11-07T14:57:01.603000-08:00",
            "tags": {}
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy2",
            "name": "sample-lifecycle-policy2",
            "status": "ENABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-09-06T10:43:21.436000-07:00",
            "dateLastRun": "2023-11-13T04:43:46.106000-08:00",
            "tags": {}
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy3",
            "name": "sample-lifecycle-policy3",
            "status": "ENABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-10-19T15:16:40.046000-07:00",
            "dateUpdated": "2023-10-21T20:07:15.958000-07:00",
            "dateLastRun": "2023-11-12T09:27:45.830000-08:00"
}]}
```

> **Note**
>
> 要使用默认值 AWS 区域，请在不带`--region`参数的情况下运行此命令。

查看生命周期策略详细信息

Image Builder 控制台中的生命周期策略详情页包含摘要部分，其他信息分组到选项卡中。页面标题为策略名称。

在 Image Builder 控制台的生命周期策略详细信息页面上，您可以查看特定生命周期策略的详细信息。您还可以在 Image Builder API 中使用命令或操作 AWS CLI 来获取策略详情。 SDKs

内容

在 Image Builder 控制台中查看生命周期策略详细信息

Image Builder 控制台中的映像详细信息页面包含摘要部分，其他信息按标签页分组。页面标题是创建映像的配方的名称和构建版本。

控制台详细信息部分和标签页

- 摘要部分
- 规则选项卡
- “范围”选项卡
- RunLog 选项卡

摘要部分

摘要部分跨越页面的宽度，包括以下详细信息。这些详细信息将始终显示。

策略状态

策略处于活动还是非活动状态。

Type

创建新映像版本（AMI 或容器映像）时 Image Builder 分配的输出映像类型。

Date created (创建日期)

生命周期策略创建以来的时间戳。

修改日期

生命周期策略上次更新的时间。

上次运行日期

生命周期策略上次运行的时间。

IAM 角色

Image Builder 用于执行生命周期操作的 IAM 角色。

ARN

生命周期策略资源的 Amazon 资源名称（ARN）。

描述

生命周期策略的描述（如果输入）。

规则选项卡

规则选项卡显示您为正在查看的策略配置的生命周期规则。该选项卡包含以下详细信息：

- 名称 – 规则的名称。基于您可以配置的策略操作，这些名称是静态的。
  - `Deprecation rule`
  - `Disable rule`
  - `Deletion rule`
- 规则 – 为规则配置的操作的简短描述。
- 规则条件 – 列出关联资源处理的配置、规则的例外情况以及保留设置（如适用）。

有关规则配置的更多信息，请参阅生命周期规则的工作方式。

“范围”选项卡

范围选项卡显示为您正在查看的策略配置的资源选择标准。该选项卡包含以下详细信息：

- Filter: ***`type of filter`***-用于定义范围的过滤器类型。筛选条件类型可以是以下类型之一：
  - `recipes` – 用于创建生命周期策略适用映像的配方。
  - `tags` – Image Builder 用于选择生命周期策略适用映像资源的一组标签。
- 搜索栏 – 您可以按名称筛选列表，以简化选项卡中显示的结果。
- 名称 – 每一行都包含您为筛选标准配置的名称或标签。
- 版本 – 如果您配置了配方筛选条件，则 Image Builder 会显示配方版本。

RunLog 选项卡

每次为配置的资源运行策略时，Image Builder 都会保存运行时详细信息。表中的每一行代表一个运行时实例。该选项卡包含以下详细信息：

- 执行 ID – 标识生命周期策略运行时实例。
- 执行状态 – 运行时状态，可报告策略操作当前是否正在运行、已成功运行、已失败或已取消。
- 受影响的资源 – 表示运行时实例是否为生命周期操作标识了任何映像资源。
- 开始日期 – 运行时实例启动时的时间戳。
- 结束日期 –运行时实例结束时的时间戳。

创建生命周期策略

创建新的 EC2 Image Builder 生命周期策略时，配置取决于该策略适用的映像类型。为 AMI 映像资源和容器镜像资源创建生命周期策略的 API 操作相同 (CreateLifecyclePolicy)。但是，映像资源和关联资源的配置有所不同。本节将介绍如何为两种资源创建生命周期管理策略。

> **Note**
>
> 创建生命周期策略之前，请确保您已满足所有 先决条件。

创建 Image Builder AMI 映像资源的生命周期管理策略

您可以使用以下方法之一通过 AWS 管理控制台 或创建 AMI 映像生命周期策略 AWS CLI。您也可以使用 CreateLifecyclePolicyAPI 操作。更多有关关联的 SDK 请求的信息，可参阅《EC2 Image Builder API 参考》中该命令的See Also链接。

AWS 管理控制台

要在中为 AMI 图像资源创建生命周期策略 AWS 管理控制台，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择生命周期策略。
3. 选择创建生命周期策略。
4. 按以下步骤中的描述配置策略设置。
5. 要在配置设置后创建生命周期策略，请选择创建策略。

配置策略的常规设置。

1. 从策略类型中选择 AMI 选项。
2. 输入策略名称。
3. （可选）输入生命周期策略的描述。
4. 默认情况下，激活处于开启状态。默认设置会激活生命周期策略并将其立即添加到计划中。要创建最初停用的策略，可以关闭激活。
5. 对于 IAM 角色，请选择以下选项之一：

   - 使用服务默认值创建生命周期执行角色-生成附有`EC2ImageBuilderLifecycleExecutionPolicy`托管策略的预配置角色。如果您没有特定的自定义权限要求，则推荐使用此选项。
   - 选择现有角色-从下拉列表中选择现有 IAM 角色。该列表会筛选出与生命周期策略执行不兼容的服务相关角色 (SLRs)。
   - 创建新的生命周期执行角色 — 打开 IAM 控制台，其中包含预先填写的信任策略和生命周期执行策略设置，便于一键创建角色。有关 step-by-step说明，请参阅为 Image Builder 生命周期管理创建 IAM 角色。

   > **Note**
   >
   > 服务相关角色与生命周期策略的执行不兼容，并且会自动从角色选择中筛选出来。如果您之前使用服务相关角色创建了生命周期策略，则可以更新该策略以使用兼容的执行角色。

配置策略的规则范围。

本部分根据您使用的筛选条件类型配置生命周期策略的资源选择。

1. 筛选条件类型：配方 – 要根据创建映像资源的配方将生命周期规则应用于映像资源，请为该策略选择最多 50 个配方版本。您可以使用分屏浏览器搜索食谱并选择特定版本。

   您还可以为语义版本使用通配符模式，通过单个策略将配方的多个版本作为目标。支持以下通配符模式：

   - `x.x.x`— 匹配食谱的所有版本。
   - `1.x.x`— 匹配主版本 1 中的所有次要版本和补丁版本。

- `1.0.x`— 匹配版本 1.0 中的所有补丁版本。

在执行时，通配符模式会解析到所有匹配的配方版本。这意味着在您设置策略后创建的新配方版本将自动包含在下次计划执行中。

> **Note**
>
> 在策略执行开始后创建或删除的配方版本要等到下次计划执行时才包括在内。

2. 筛选条件类型：标签 – 要根据资源标签将生命周期规则应用于映像资源，请输入最多包含 50 个键值对的列表，供策略进行匹配。Image Builder 通过标签扫描发现资源，并将生命周期规则应用于匹配的资源。

> **Note**
>
> 策略执行开始后标记或未标记的资源要等到下次计划执行时才包括在内。

为您的策略配置排除规则（可选）。

排除规则定义生命周期规则的例外情况。符合排除规则所指定标准的资源将被排除在生命周期操作之外。

1. 对于 AMI 映像策略，您可以配置以下 AMI 排除规则：
   - 排除公众 AMIs-选择此选项可将公众排除在生命周期操作 AMIs 之外。
   - AMIs 按区域排除-指定 AWS 区域 从生命周期操作中排除。
   - 排除与账户 AMIs 共享-指定生命周期操作中 AMIs 应排除 AWS 账户 谁的共享。
   - 排除最近启动的实例 AMIs-指定 AMIs 要排除最近用于启动实例的时间段。
   - AMIs 按标签排除-指定 AMIs 应从生命周期操作中排除的标签。
2. 对于基于标签的 Image Builder 图像策略，您可以为生命周期操作中应排除的 Image Builder 图像资源指定标签。

启用以下一条或多条生命周期规则，以应用于生命周期策略选择的资源。如果某个资源在策略运行时与多个生命周期规则相匹配，则 Image Builder 将按以下顺序执行规则操作：1) 弃用、2) 禁用、3) 删除。

弃用规则

将 Image Builder 映像资源状态设置为 Deprecated。对于已弃用的映像，Image Builder 管道仍会运行。您可以选择为关联设置弃用时间， AMIs 而不会影响您启动新实例的能力。

- 单位数 – 指定映像资源创建后必须经过的时间段的整数值，然后将其标记为 Deprecated。
- 单位 – 选择要使用的时间范围。该值可以为 Days、Weeks、Months 或 Years。
- 弃用 AMIs — 选中该复选框可将关联的 Amazon EC2 AMIs 标记为弃用日期。它们 AMIs 仍然可用，您仍然可以从中启动新实例。

禁用规则

将 Image Builder 映像资源状态设置为 Disabled。对于该映像，此操作会阻止 Image Builder 管道运行。您可以选择禁用关联的 AMI 以防止启动新实例。

- 单位数 – 指定映像资源创建后必须经过的时间段的整数值，然后将其标记为 Disabled。
- 单位 – 选择要使用的时间范围。该值可以为 Days、Weeks、Months 或 Years。
- 禁用 AMIs-选中该复选框可禁用关联的 Amazon EC2 AMIs。您不能再使用它们 AMIs 或从中启动新实例。

删除规则

按使用期限或数量删除映像资源。您可以定义满足您需求的阈值。当 Image Builder 映像资源超过阈值时，会将其删除。您可以选择取消注册关联的快照， AMIs 也可以删除这些 AMIs快照。您还可以为希望保留超过阈值的资源指定标签。

按使用期限配置删除规则时，Image Builder 会在您配置的一段时间后删除映像资源。例如，在 6 个月后删除映像资源。按数量配置时，Image Builder 会保留您指定的最新映像数量或尽可能接近该数量，并删除早期版本。

- 按使用期限
  - 单位数 – 指定映像资源创建后必须经过的时间段的整数值，然后将其删除。
  - 单位 – 选择要使用的时间范围。该值可以为 Days、Weeks、Months 或 Years。
  - 每个配方至少保留一个映像 – 选中该复选框，为受此规则影响的每个配方版本保留最新的可用映像资源。

按数量

- 映像计数 – 指定为每个配方版本保留的最新映像资源数量的整数值。

- 取消注册 AMIs — 选中该复选框可取消注册关联的 Amazon EC2。 AMIs您不能再使用它们 AMIs 或从中启动新实例。
- 保留带有关联标签的图像和快照-选中该复选框可输入要保留的图像资源的标签列表。 AMIs标签适用于图像资源和 Amazon EC2 AMIs。最多可输入 50 个键值对。

标签（可选）

将标签添加到生命周期策略。

AWS CLI

要创建新的 Image Builder 生命周期策略，您可以在 AWS CLI中使用 create-lifecycle-policy 命令。

创建 Image Builder 容器映像资源的生命周期管理策略

您可以使用以下方法之一通过 AWS 管理控制台 或创建容器映像生命周期策略 AWS CLI。您也可以使用 CreateLifecyclePolicyAPI 操作。更多有关关联的 SDK 请求的信息，可参阅《EC2 Image Builder API 参考》中该命令的See Also链接。

AWS 管理控制台

要在中为容器镜像资源创建生命周期策略 AWS 管理控制台，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择生命周期策略。
3. 选择创建生命周期策略。
4. 按以下步骤中的描述配置策略设置。
5. 要在配置设置后创建生命周期策略，请选择创建策略。

策略配置：常规设置

配置策略的常规设置。

1. 从 “策略类型” 中选择 “容器镜像” 选项。

2. 输入策略名称。
3. （可选）输入生命周期策略的描述。
4. 默认情况下，激活处于开启状态。默认设置会激活生命周期策略并将其立即添加到计划中。要创建最初停用的策略，可以关闭激活。
5. 对于 IAM 角色，请选择以下选项之一：

   - 使用服务默认值创建生命周期执行角色-生成附有`EC2ImageBuilderLifecycleExecutionPolicy`托管策略的预配置角色。如果您没有特定的自定义权限要求，则推荐使用此选项。
   - 选择现有角色-从下拉列表中选择现有 IAM 角色。该列表会筛选出与生命周期策略执行不兼容的服务相关角色 (SLRs)。
   - 创建新的生命周期执行角色 — 打开 IAM 控制台，其中包含预先填写的信任策略和生命周期执行策略设置，便于一键创建角色。有关 step-by-step说明，请参阅为 Image Builder 生命周期管理创建 IAM 角色。

   > **Note**
   >
   > 服务相关角色与生命周期策略的执行不兼容，并且会自动从角色选择中筛选出来。如果您之前使用服务相关角色创建了生命周期策略，则可以更新该策略以使用兼容的执行角色。

配置策略的规则范围。

本部分根据您使用的筛选条件类型配置生命周期策略的资源选择。

1. 筛选条件类型：配方 – 要根据创建映像资源的配方将生命周期规则应用于映像资源，请为该策略选择最多 50 个配方版本。您可以使用分屏浏览器搜索食谱并选择特定版本。

   您还可以为语义版本使用通配符模式，通过单个策略将配方的多个版本作为目标。支持以下通配符模式：

   - `x.x.x`— 匹配食谱的所有版本。
   - `1.x.x`— 匹配主版本 1 中的所有次要版本和补丁版本。
   - `1.0.x`— 匹配版本 1.0 中的所有补丁版本。

在执行时，通配符模式会解析到所有匹配的配方版本。这意味着在您设置策略后创建的新配方版本将自动包含在下次计划执行中。

> **Note**
>
> 在策略执行开始后创建或删除的配方版本要等到下次计划执行时才包括在内。

2. 筛选条件类型：标签 – 要根据资源标签将生命周期规则应用于映像资源，请输入最多包含 50 个键值对的列表，供策略进行匹配。Image Builder 通过标签扫描发现资源，并将生命周期规则应用于匹配的资源。

> **Note**
>
> 策略执行开始后标记或未标记的资源要等到下次计划执行时才包括在内。

为您的策略配置排除规则（可选）。

排除规则定义生命周期规则的例外情况。符合排除规则所指定标准的资源将被排除在生命周期操作之外。

1. 对于 AMI 映像策略，您可以配置以下 AMI 排除规则：
   - 排除公众 AMIs-选择此选项可将公众排除在生命周期操作 AMIs 之外。
   - AMIs 按区域排除-指定 AWS 区域 从生命周期操作中排除。
   - 排除与账户 AMIs 共享-指定生命周期操作中 AMIs 应排除 AWS 账户 谁的共享。
   - 排除最近启动的实例 AMIs-指定 AMIs 要排除最近用于启动实例的时间段。
   - AMIs 按标签排除-指定 AMIs 应从生命周期操作中排除的标签。
2. 对于基于标签的 Image Builder 图像策略，您可以为生命周期操作中应排除的 Image Builder 图像资源指定标签。

删除规则

对于容器映像，此规则会删除 Image Builder 容器映像资源。您可以选择删除已分配到 ECR 存储库的 Docker 映像，以防止将其用于运行新的容器。

按使用期限配置删除规则时，Image Builder 会在您配置的一段时间后删除映像资源。例如，在 6 个月后删除映像资源。按数量配置时，Image Builder 会保留您指定的最新映像数量或尽可能接近该数量，并删除早期版本。

- 按使用期限
  - 单位数 – 指定映像资源创建后必须经过的时间段的整数值，然后将其删除。
  - 单位 – 选择要使用的时间范围。该值可以为 `Days`、`Weeks`、`Months` 或 `Years`。
  - 至少保留一个映像 – 选中该复选框，为受此规则影响的每个配方版本仅保留最新的可用映像资源。

  按数量
  - 映像计数 – 指定为每个配方版本保留的最新映像资源数量的整数值。
- 删除 ECR 容器映像 – 选中该复选框，删除存储在 ECR 存储库中关联的容器映像。无法再将容器映像作为创建新映像或运行新容器的基础。
- 保留具有关联标签的映像 – 选中该复选框，输入要保留的映像资源的标签列表。

标签（可选）

将标签添加到生命周期策略。

AWS CLI

要创建新的 Image Builder 生命周期策略，您可以在 AWS CLI中使用 create-lifecycle-policy 命令。

生命周期管理规则如何适用于 Image Builder 映像资源

映像生命周期策略使用您定义的生命周期规则来实施您的整体资源管理策略。您定义的规则有助于确保可用映像的新鲜度，并最大限度地降低底层基础架构的成本，例如用于输出的 AMIs快照存储或容器映像的 ECR 存储库存储和数据传输速率。

您可以为您的策略配置以下类型的规则。

弃用规则

将 Image Builder 映像资源状态设置为 `Deprecated`。对于已弃用的映像，Image Builder 管道仍会运行。您可以选择为关联设置弃用时间， AMIs 而不会影响您启动新实例的能力。

弃用某个 AMI 时，常规搜索会将其忽略。例如，如果您在中运行 Amazon EC2 describe-images 命令 AWS CLI，则该命令不会AMIs 在结果集中返回已弃用。但是，您仍然可以发现他们的 AMI ID 已被弃用 AMIs 。

此规则不适用于容器映像。

禁用规则

将 Image Builder 映像资源状态设置为 `Disabled`。对于该映像，此操作会阻止 Image Builder 管道运行。您可以选择禁用关联的 AMI 以防止启动新实例。

禁用某个 AMI 之后，其便会变为私有，无法用于启动新实例。如果您与任何账户、组织或组织单位共享 AMI，则当您的 AMI 变为私有时，他们将失去对该 AMI 的访问权限。

此规则不适用于容器映像。

删除规则

按使用期限或数量删除映像资源。您可以定义满足您需求的阈值。当 Image Builder 映像资源超过阈值时，会将其删除。您可以选择取消注册关联的快照， AMIs 也可以删除这些 AMIs快照。您还可以为希望保留超过阈值的资源指定标签。

对于容器映像，此规则会删除 Image Builder 容器映像资源。您可以选择删除已分配到 ECR 存储库的容器映像，以防止将其用于运行新的容器。

内容

AMI 生命周期排除规则

以下排除规则定义了生命周期规则的例外情况 AMIs。 AMIs 符合排除规则所指定标准的将被排除在生命周期操作之外。您可以使用 API 和在 AWS 管理控制台 或中配置排除规则 AWS CLI。

以下术语使用 `LifecyclePolicyDetailExclusionRules` 数据类型中的 API 表示法。

排除规则

ami

包含 `LifecyclePolicyDetailExclusionRulesAmis` 中的设置，如下面的列表所示。

tagMap

您可以提供多达 50 个标签的列表，这些标签可跳过任何类型资源的生命周期操作。

以下术语使用 LifecyclePolicyDetailExclusionRulesAmis 数据类型中的 API 表示法。

AMI 排除规则

isPublic

配置是否将公众排除 AMIs 在生命周期操作之外。

lastLaunched

指定 Image Builder 的配置详细信息，以从生命周期操作中排除最新资源。

区域

从生命周期操作中排除的配置 AWS 区域 。

sharedAccounts

指定 AWS 账户 将哪些资源排除在生命周期操作之外。

tagMap

列出应从包含标签的生命周期操作中排除 AMIs 的标签。

查看策略的生命周期管理规则详细信息

规则是在您为 Image Builder 映像资源创建的生命周期管理策略中定义的。在控制台中，生命周期策略详细信息页面中有一个 规则选项卡，显示了您为策略配置的规则的详细信息。

要在中获取策略详细信息 AWS CLI，可以运行get-lifecycle-policy命令。响应中的策略详细信息包含为策略定义的操作（规则）列表，其中包括已配置的所有设置。

使用 Image Builder 配置自定义映像

配置资源是构成映像管道的构建基块，也是这些管道生成的映像。本章介绍创建、维护和共享 Image Builder 资源，包括组件、配方和映像，以及基础架构配置和分配设置。

> **Note**
>
> 为了帮助您管理 Image Builder 资源，您可通过标签 的形式为每个资源分配元数据。您使用标签按照不同方式（例如按用途、所有者或环境）对 AWS 资源进行分类。这在您有许多相同类型的资源时会非常有用。您可以根据分配到特定资源的标签来更轻松地识别该资源。
> 有关使用 Image Builder 命令为资源添加标签的更多信息 AWS CLI，请参阅本指南的标记资源章节。

内容

在 Image Builder 中管理配方

EC2 Image Builder 配方定义了用作创建新图像的起点的基础图像，以及为自定义图像和验证一切是否按预期运行而添加的一组组组件。Image Builder 为每个组件提供版本自动选择。默认情况下，您最多可以对配方应用 20 个组件。这包括构建和测试组件。

在您创建配方后，您不能再修改或替换它。要在创建配方后更新组件，您必须创建新的配方或配方版本。您可以将标签应用到现有配方。有关使用 Image Builder 命令为资源添加标签的更多信息 AWS CLI，请参阅本指南的标记资源部分。

> **Tip**
>
> 您可以在配方中使用 Amazon 托管组件，也可以开发自己的自定义组件。有关更多信息，请参阅 为 Image Builder 映像开发自定义组件。对于创建输出的图像配方 AMIs，您也可以使用 AWS Marketplace 图片产品和组件。有关与 AWS Marketplace 产品集成的更多信息，请参阅AWS Marketplace 在 Image Builder 中集成。

本节介绍如何列出、查看和创建配方。

内容

列出和查看映像配方的详细信息

本节介绍您可以通过多种方式查找有关您的 EC2 Image Builder 映像配方的信息并查看详细信息。

映像配方详细信息

通过控制台列出映像配方

要在 Image Builder 控制台中查看在您账户下创建的映像配方列表，请按照以下步骤操作：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择映像配方。这会显示在您账户下创建的映像配方列表。
3. 要查看详细信息或创建新的配方版本，请选择配方名称链接。这将打开配方的详细视图。

> **Note**
>
> 您也可以选中配方名称 旁边的复选框，然后选择 查看详细信息。

列出来自 AWS CLI

以下示例说明了如何使用 AWS CLI列出所有映像配方。

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

通过控制台查看映像配方详细信息

要使用 Image Builder 控制台查看特定映像配方的详细信息，请按照 通过控制台列出映像配方 中所述的步骤选择要查看的映像配方。

在配方详细信息页面上，您可以：

- 删除配方。有关在 Image Builder 中删除资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。
- 创建新版本。
- 从配方中创建管道。选择根据此配方中创建管道后，您将进入管道向导。有关使用管道向导创建 Image Builder 管道的更多信息，请参阅 教程：通过 Image Builder 控制台向导使用输出 AMI 创建映像管道

> **Note**
>
> 当您根据现有配方创建管道时，创建新配方的选项不可用。

从中获取图片配方详情 AWS CLI

以下示例说明了如何指定 Amazon 资源名称（ARN）以通过使用 imagebuilder CLI 命令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/2020.12.03
```

从中获取图片配方政策详情 AWS CLI

以下示例说明了如何指定 ARN 以通过使用 imagebuilder CLI 命令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/2020.12.03
```

列出和查看容器配方详细信息

本节介绍如何查找有关 EC2 Image Builder 容器配方的信息并查看详细信息。

容器配方详细信息

- 在控制台中列出容器配方
- 使用列出容器配方 AWS CLI
- 在控制台中查看容器配方详细信息
- 使用获取容器配方详情 AWS CLI
- 通过获取容器配方政策的详细信息 AWS CLI

在控制台中列出容器配方

要在 Image Builder 控制台中查看在您的账户下创建的容器配方列表，请按照以下步骤操作：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择容器配方。这将显示在您的账户下创建的容器配方列表。
3. 要查看详细信息或创建新的配方版本，请选择配方名称链接。这将打开配方的详细视图。

> **Note**
>
> 您也可以选中配方名称旁边的框，然后选择查看详细信息。

使用列出容器配方 AWS CLI

以下示例说明了如何使用 AWS CLI列出您的所有容器配方。

```
aws imagebuilder list-container-recipes
```

在控制台中查看容器配方详细信息

要使用 Image Builder 控制台查看特定容器配方的详细信息，请选择要查看的容器配方，然后执行 在控制台中列出容器配方 中所述的步骤。

在配方详细悉心页面上，您可以执行以下操作：

- 删除配方。有关如何在 Image Builder 中删除资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。
- 创建新版本。
- 从配方中创建管道。选择从此配方中创建管道后，您将进入管道向导。有关如何使用管道向导创建 Image Builder 管道的更多信息，请参阅 教程：通过 Image Builder 控制台向导使用输出 AMI 创建映像管道

> **Note**
>
> 当您从现有配方中创建管道时，创建新配方的选项不可用。

使用获取容器配方详情 AWS CLI

以下示例说明了如何使用 imagebuilder CLI 命令通过指定其 ARN 来获取容器配方的详细信息。

```
aws imagebuilder get-container-recipe --container-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:container-recipe/my-example-recipe/2020.12.03
```

通过获取容器配方政策的详细信息 AWS CLI

以下示例说明了如何使用 imagebuilder CLI 命令通过指定其 ARN 来获取容器配方策略的详细信息。

```
aws imagebuilder get-container-recipe-policy --container-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-example-
recipe/2020.12.03
```

创建映像配方的新版本

本节介绍如何创建映像配方的新版本。

内容

- 通过控制台创建新的映像配方版本
- 使用创建图像配方 AWS CLI
- 在控制台中导入虚拟机 (VM) 作为基础映像

通过控制台创建新的映像配方版本

当您创建新的配方版本时，它与创建新配方的操作几乎相同。不同之处在于，在大多数情况下，为匹配基础配方，某些详细信息都是预先选择的。以下列表描述创建新配方和创建现有配方的新版本之间的区别。

新版本中的基础配方详细信息

- 名称 - 不可编辑。
- 版本 - 必填。输入要创建的版本号，格式为 <major>.<minor>.<patch>。Image Builder 支持食谱的自动版本递增，允许您在配方版本中使用通配符模式。使用通配符版本（例如）创建配方时`1.0.x`，Image Builder 会自动增加版本（例如`1.0.11.0.2`、`1.0.3`、等）。这样就无需手动跟踪和增加配方版本。
- 选择映像选项 - 已预先选择，但您可以对其进行编辑。如果您更改对基础映像来源的选择，则可能会丢失其他详细信息，这些详细信息取决于您选择的原始选项。

  要查看与您的基础映像选择相关的详细信息，请选择与您的选择相匹配的选项卡。

  Managed image

  - 映像操作系统 (OS) - 不可编辑。
  - 映像名称 - 根据您为现有配方所做的基础映像选择的组合进行预先选择。但是，如果您更改选择映像选项，则会丢失预先选择的映像名称。
  - 自动版本控制选项 - 与您的基础配方不匹配。此映像选项默认为使用选定的操作系统版本选项。

  > **Important**
  >
  > 如果您使用语义版本控制来启动管道构建，请确保将此值更改为使用最新的可用操作系统版本。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

  AWS Marketplace image

  - 订阅 — 此选项卡应处于打开状态，并且 AWS Marketplace 应预先选择订阅的图片以匹配您的基本食谱。如果您更改配方用作基础映像的映像，则可能会丢失其他详细信息，这些详细信息取决于您选择的原始映像。

  有关 AWS Marketplace 产品的更多信息，请参阅《买AWS Marketplace 家指南》中的 “购买商品”。

Custom AMI

AMI 来源（必填）-输入 AMI ID 或 AWS Systems Manager (SSM) 参数存储参数，其中包含用作基础映像的 AMI ID。SSM 代理必须预先安装在选定的 AMI 中。

- AMI ID — 此设置未预先填入您的原始条目。输入您的基础映像的 AMI ID。示例：*ami-1234567890abcdef1*。
- SSM 参数-输入包含基础映像的 AMI ID 的 SSM 参数存储参数的名称或 ARN。示例：*/ib/test/param* 或 `arn:aws:ssm:`*us-east-1*`:`*111122223333*`:parameter`*/ib/test/param*。

- 实例配置 - 设置是预先选择的，但您可以对其进行编辑。
  - Systems Manager 代理 – 您可以选中或清除此复选框来控制在新映像上安装 Systems Manager 代理。默认情况下，该复选框处于清除状态，以便在新映像中包含 Systems Manager 代理。要从最终映像中移除 Systems Manager 代理，请选中该复选框以使代理不包含在您的 AMI 中。
  - 用户数据 – 当您启动构建实例时，可以使用此区域以提供要运行的命令或命令脚本。但是，此值会替换 Image Builder 可能添加的任何命令，以确保安装 Systems Manager。这些命令包括 Image Builder 通常在创建新映像之前为 Linux 映像运行的清理脚本。

    当 Image Builder 启动实例时，用户数据脚本将在组件执行开始之前的云初始化阶段运行。此步骤将记录到实例上的以下文件中：`var/log/cloud-init.log`。

    > **Note**
    > - 如果您要输入用户数据，请确保在您的基础映像上预先安装 Systems Manager 代理，或者将安装的内容包含在您的用户数据中。
    > - 对于 Linux 映像，请通过创建一条在用户数据脚本中命名为 `perform_cleanup` 的空文件的命令，以确保运行清理步骤。Image Builder 会检测到此文件，并在创建新映像之前运行清理脚本。有关更多信息和示例脚本，请参阅 Image Builder 的安全最佳实践。

- 工作目录 - 已预先选择，但您可以对其进行编辑。
- 组件 - 已包含在配方中的组件显示在每个组件列表（构建和测试）末尾的选定组件部分中。您可以移除所选组件或对其重新排序，以满足您的需要。

  CIS 加固组件未遵循 Image Builder 配方中的标准组件排序规则。CIS 强化组件始终最后运行，以确保基准测试针对您的输出映像运行。

> **Note**
>
> 构建和测试组件列表根据组件所有者类型显示可用组件。要添加组件，请选择添加构建组件，然后选择适用的所有权筛选器。例如，要添加与 AWS Marketplace 产品关联的生成组件，请选择`AWS Marketplace`。这将在控制台界面的右侧打开一个列出 AWS Marketplace 组件的选择面板。
> 对于 CIS 组件，请选择`Third party managed`。

您可以为自己的所选组件配置以下设置：

- 版本控制选项 - 已预先选择，但您可以对其进行更改。我们建议您选择使用最新的可用组件版本选项，以确保您的映像版本始终使用最新版本的组件。如果您需要在配方中使用特定的组件版本，则可以选择指定组件版本，然后在出现的组件版本框中输入版本。
- 输入参数 - 显示组件接受的输入参数。该值预先填充了配方先前版本中的值。如果您在此配方中首次使用此组件，并且已为该输入参数定义了默认值，则默认值以灰色文本显示在值框中。如果未输入其他值，Image Builder 将使用默认值。

  如果需要输入参数，但组件中未定义默认值，则必须提供一个值。如果缺少任何必需的参数且未定义默认值，Image Builder 将不会创建配方版本。

  > **Important**
  >
  > 组件参数是纯文本值，并且已记录在 AWS CloudTrail中。我们建议您使用 AWS Secrets Manager 或 P AWS Systems Manager arameter Store 来存储您的密钥。有关 Secrets Manager 的更多信息，请参阅AWS Secrets Manager 用户指南中的什么是 Secrets Manager?。有关 AWS Systems Manager Parameter Store 的更多信息，请参阅《AWS Systems Manager 用户指南》中的AWS Systems Manager Parameter Store。

要展开版本控制选项或输入参数的设置，可以选择设置名称旁边的箭头。要展开所有选定组件的所有设置，可以关闭和打开全部展开开关。

- 存储（卷）- 已预先填好。根卷设备名称、快照和 IOPS 选项不可编辑。但是，您可以更改所有其余设置，例如大小。您还可以添加新卷以及加密新卷或现有卷。

  要在源区域（运行构建的地方）为 Image Builder 在您的账户下创建的映像进行加密，您必须在映像配方中使用存储卷加密。在该构建的分配阶段运行的加密仅适用于分配给其他账户或地区的映像。

> **Note**
>
> 如果您对卷使用加密，则必须分别为每个卷选择密钥，即使该密钥与用于根卷的密钥相同也是如此。

创建新的映像配方版本：

1. 在配方详细信息页面顶部，选择创建新版本。这将带您进入创建映像配方页面。
2. 要创建新版本，请进行更改，然后选择Create recipe（创建配方）。

   您的最终图片最多可以包含来自 AWS Marketplace 图片产品和组件的四个产品代码。如果您选择的基本图片和组件包含四个以上的产品代码，则当您尝试创建配方时，Image Builder 会返回错误。

有关如何在创建映像管道时创建映像配方的更多信息，请参阅本指南的入门部分的 步骤 2：选择配方。

使用创建图像配方 AWS CLI

要使用中的 Image Builder `create-image-recipe` 命令创建图像配方 AWS CLI，请执行以下步骤：

先决条件

在运行本节中的 Image Builder 命令从中创建图像配方之前 AWS CLI，您可以选择创建配方使用的组件。以下步骤中的映像配方示例引用了在本指南 从中创建自定义组件 AWS CLI 部分中创建的示例组件。

如果要在食谱中添加组件，请 ARNs 记下要包含的组件。您也可以创建没有任何组件的配方，用于测试现有 AMIs 或仅限分发的工作流程。

1. 创建 CLI 输入 JSON 文件

   您可以使用内联命令参数为 create-image-recipe 命令提供所有输入。但是，生成的命令可能会很长。为了简化命令，您可以改为提供包含所有配方设置的 JSON 文件。

> **Note**
>
> JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 操作请求参数，请参阅《EC2 Image Builder API 参考》中的CreateImageRecipe命令。
> 要将数据值作为命令行参数提供，请参阅《AWS CLI 命令引用》中指定的参数名称。

以下是这些示例指定的参数的摘要：

- 名称（字符串，必填项）- 映像配方的名称。
- 描述（字符串）- 映像配方的描述。
- parentImage （字符串，必填项）- 映像配方用作自定义映像基础的映像。您可以使用以下选项之一来指定父映像：
  - AMI ID
  - Image Builder 图片资源 ARN
  - AWS Systems Manager (SSM) 参数存储参数，前缀为`ssm:`，后跟参数名称或 ARN。
  - AWS Marketplace 产品编号

  > **Note**
  >
  > Linux 和 macOS 示例使用 Image Builder AMI，Windows 示例使用 ARN。

- SemanticVersion <major>（字符串，必填）-输入要按格式创建的版本号。 <minor>。<patch>。Image Builder 支持食谱的自动版本递增，允许您在配方版本中使用通配符模式。使用通配符版本（例如）创建配方时`1.0.x`，Image Builder 会自动增加版本（例如`1.0.11.0.2`、`1.0.3`、等）。这样就无需手动跟踪和增加配方版本。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。
- 组件（数组，可选）-包含`ComponentConfiguration`对象数组。组件是可选的-您可以创建没有任何组件的配方，用于测试或分发工作流程：

  > **Note**
  >
  > Image Builder 按照您在配方中指定的顺序安装组件。但是，CIS 强化组件始终最后运行，以确保基准测试针对您的输出映像运行。

- componentARN（字符串，必填）– 组件 ARN。

  > **Tip**
  >
  > 要使用其中一个示例来创建自己的图像配方，必须将该示例 ARNs 替换 ARNs 为用于配方的组件。

- 参数（对象数组）- 包含 `ComponentParameter` 对象数组。如果需要输入参数，但组件中未定义默认值，则必须提供一个值。如果缺少任何必需的参数且未定义默认值，Image Builder 将不会创建配方版本。

  > **Important**
  >
  > 组件参数是纯文本值，并且已记录在 AWS CloudTrail中。我们建议您使用 AWS Secrets Manager 或 P AWS Systems Manager arameter Store 来存储您的密钥。有关 Secrets Manager 的更多信息，请参阅AWS Secrets Manager 用户指南中的什么是 Secrets Manager?。有关 AWS Systems Manager Parameter Store 的更多信息，请参阅《AWS Systems Manager 用户指南》中的AWS Systems Manager Parameter Store。

  - 名称（字符串，必填）- 要设置的组件参数的名称。
  - 值（字符串数组，必填）- 包含用于设置指定组件参数值的字符串数组。如果为组件定义了默认值，但未提供其他值，则 AWSTOE 使用默认值。

- additionalInstanceConfiguration(object)-为您的编译实例指定其他设置并启动脚本。
  - systemsManagerAgent（对象）-包含编译实例上的 Systems Manager 代理的设置。
    - uninstallAfterBuild（布尔值）-控制在创建新 AMI 之前是否从最终构建映像中删除 Systems Manager 代理。如果将此选项设置为 `true`，则从最终映像中移除该代理。如果将此选项设置为 `false`，则保留该代理，以便将它包含在新 AMI 中。默认值为 `false`。

      > **Note**
      >
      > 如果 JSON 文件中未包含该 `uninstallAfterBuild` 属性，并且满足以下条件，则 Image Builder 会从最终映像中移除 Systems Manager 代理，使其在 AMI 中不可用：

- userDataOverride 为空或已从 JSON 文件中省略。
- 对于未在基础映像上预装代理的操作系统，Image Builder 会自动在构建实例上安装 Systems Manager 代理。

- userDataOverride（字符串）-提供启动构建实例时要运行的命令或命令脚本。

> **Note**
>
> 用户数据始终采用 base 64 编码。例如，以下命令被编码为 IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg==：
>
> ```
> #!/bin/bash
> mkdir -p /var/bb/
> touch /var
> ```
>
> Linux 示例使用此编码值。

Linux

以下示例中的基础映像（parentImage 属性）是 AMI。使用 AMI 时，您必须具有访问该 AMI 的权限，并且 AMI 必须位于源区域（与 Image Builder 运行命令的区域相同）。将文件另存为 create-image-recipe.json，并在 create-image-recipe 命令中使用。

```
{
"name": "BB Ubuntu Image recipe",
"description": "Hello World image recipe for Linux.",
"parentImage": "ami-1234567890abcdef1",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/bb$"
 }
],
"additionalInstanceConfiguration": {
 "systemsManagerAgent": {
   "uninstallAfterBuild": true
 },
 "userDataOverride": "IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg=="
}
```

```
}
```

Windows

以下示例引用最新版本的 Windows Server 2016 英文版全基础映像。此示例中的 ARN 引用基于您指定的语义版本过滤器的最新图像:. `arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/x.x.x`

```
{
"name": "MyBasicRecipe",
"description": "This example image recipe creates a Windows 2016 image.",
"parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/x.x.x",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/my-example-component/2019.12.02/1"
 },
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/my-imported-component/1.0.0/1"
 }
]
}
```

> **Note**
>
> 要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

macOS

以下示例中的基础映像（`parentImage` 属性）是 AMI。使用 AMI 时，您必须具有访问该 AMI 的权限，并且 AMI 必须位于源区域（与 Image Builder 运行命令的区域相同）。将文件另存为 `create-image-recipe.json`，并在 create-image-recipe 命令中使用。

```
{
"name": "macOS Catalina Image recipe",
```

```
"description": "Hello World image recipe for macOS.",
"parentImage": "ami-1234567890abcdef1",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-
west-2:111122223333:component/catalina$"
 }
],
"additionalInstanceConfiguration": {
 "systemsManagerAgent": {
   "uninstallAfterBuild": true
 },
 "userDataOverride": "IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg=="
}
}
```

示例：不含组件的食谱

您可以创建没有任何组件的配方，用于测试现有 AMIs 或仅限分发的工作流程。以下示例显示了一个使用现有 AMI 而不应用任何其他组件的配方：

```
{
 "name": "Test Distribution Recipe",
 "description": "Recipe for testing and distributing existing AMI without
 modifications.",
 "parentImage": "ami-1234567890abcdef1",
 "semanticVersion": "1.0.0",
 "additionalInstanceConfiguration": {
  "systemsManagerAgent": {
    "uninstallAfterBuild": true
  }
 }
}
```

2. 创建配方

使用以下命令以创建配方。在 `--cli-input-json` 参数中提供您在上一步中创建的 JSON 文件的名称：

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

您的最终图片最多可以包含来自 AWS Marketplace 图片产品和组件的四个产品代码。如果您选择的基础图片和组件包含四个以上的产品代码，则在运行`create-image-recipe`命令时，Image Builder 会返回错误。

在控制台中导入虚拟机 (VM) 作为基础映像

在本节中，我们将重点介绍如何导入虚拟机 (VM) 作为映像配方的基础映像。这里我们不介绍创建配方或配方版本所涉及的其他步骤。有关在 Image Builder 控制台中使用管道创建向导创建新映像配方的其他步骤，请参阅 管道向导：创建 AMI。有关创建新映像配方或配方版本的其他步骤，请参阅 创建映像配方的新版本。

要在 Image Builder 控制台中导入虚拟机 (VM) 作为映像配方的基础映像，请按照以下步骤以及任何其他必需的步骤来创建您的配方或配方版本。

1. 在基础映像的选择映像部分中，选择导入基础映像选项。
2. 像往常一样选择映像操作系统 (OS)和操作系统 (OS) 版本。

虚拟机导入配置

当您从虚拟化环境中导出虚拟机时，该过程会创建一个或多个磁盘容器文件，这些文件充当虚拟机环境、设置和数据的快照。您可以使用这些文件导入虚拟机作为映像配方的基本映像。有关 VMs 在 Image Builder 中导入的更多信息，请参阅 导入和导出 VM 映像

要指定导入源的位置，请执行以下步骤：

导入源

在磁盘容器 1 部分中指定要导入的第一个虚拟机映像磁盘容器或快照的来源。

1. 源 – 可以是 S3 存储桶或 EBS 快照。
2. 选择磁盘的 S3 位置 – 输入 Amazon S3 中存储磁盘映像的位置。要浏览位置，请选择浏览 S3。
3. 要添加磁盘容器，请选择添加磁盘容器。

IAM 角色

要将 IAM 角色与您的虚拟机导入配置相关联，请从 IAM 角色下拉列表中选择该角色，或者选择创建新角色来创建一个新角色。如果您创建了新角色，IAM 角色控制台页面将在单独的标签页中打开。

高级设置 – 可选

以下设置可选：使用这些设置，您可以为导入创建的基本映像配置加密、许可、标签等。

General

1. 为基本映像指定一个唯一的名称。如果不输入任何值，则基础映像将继承该配方名称。
2. 指定基础映像的版本。采用以下格式：*<major>.<minor>.<patch>*。如果不输入任何值，则基础映像将继承该配方版本。
3. 您也可以为基础映像输入描述。

基础映像架构

要指定虚拟机导入源的架构，请从架构列表中选择一个值。

加密

如果您的虚拟机磁盘映像已加密，则必须提供用于导入过程的密钥。要 AWS KMS key 为导入指定一个，请从加密（KMS 密钥）列表中选择一个值。该列表包含您的账户在当前区域中有权访问的 KMS 密钥。

许可证管理

导入虚拟机时，导入过程会自动检测虚拟机操作系统并将相应的许可证应用于基本映像。根据您的操作系统平台，许可证类型如下：

- 包含许可证 — 适用于您的平台的相应 AWS 许可证将应用于您的基本映像。
- 自带许可（BYOL） - 保留源自虚拟机的许可证（如果适用）。

要将使用创建的许可证配置附加 AWS License Manager 到您的基础映像，请从许可证配置名称列表中进行选择。有关 License Manager 的更多信息，请参阅使用 AWS License Manager

> **Note**
>
> - 许可证配置包含基于您的企业协议条款的许可规则。
> - Linux 仅支持 BYOL 许可证。

标签（基本映像）

标签使用键值对为您的 Image Builder 资源分配可搜索的文本。要为导入的基础映像指定标签，请使用键和值框输入键值对。

要添加标签，请选择 Add tag（添加标签）。要删除标签，请选择 Remove tag（删除标签）。

创建新版本的容器配方

本节展示了如何创建创建新版本的容器配方。

内容

- 使用控制台创建新的容器配方版本
- 使用创建容器配方 AWS CLI

使用控制台创建新的容器配方版本

创建新版本的容器配方与创建新配方几乎相同。不同之处在于，在大多数情况下，为匹配基础配方，某些详细信息都是预先选择的。以下列表描述创建新配方和创建现有配方的新版本之间的区别。

配方详细信息

- 名称 - 不可编辑。
- 版本 - 必填。此详细信息未预先填充当前版本或任何类型的序列。以 major.minor.patch 格式输入要创建的版本号。如果该版本已经存在，则会遇到错误。

基础映像

- 选择映像选项 - 预先选择，但可以编辑。如果您更改对基础映像来源的选择，则可能会丢失其他详细信息，这些详细信息取决于您选择的原始选项。

  对于 Docker 容器镜像，您可以从托管的公共镜像 DockerHub、Amazon ECR 中的现有容器镜像或亚马逊管理的容器镜像中进行选择。要查看与您的基础映像选择相关的详细信息，请选择与您的选择相匹配的选项卡。

  Managed images

  - 映像操作系统 (OS) - 不可编辑。
  - 映像名称 - 根据您为现有配方所做的基础映像选择的组合进行预先选择。但是，如果您更改选择映像选项，则会丢失预先选择的映像名称。
  - 自动版本控制选项 - 与您的基础配方不匹配。自动版本控制选项默认为使用选定的操作系统版本选项。

    > **Important**
    >
    > 如果您使用语义版本控制来启动管道构建，请确保将此值更改为使用最新的可用操作系统版本。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。

  ECR image

  - 映像操作系统 (OS) - 预先选择，但可编辑。
  - 操作系统版本 - 预先选择，但可编辑。
  - ECR 映像 ID – 已预先填充，但可编辑。

  Docker Hub image

  - 映像操作系统 (OS) - 不可编辑。
  - 操作系统版本 - 预先选择，但可编辑。
  - Docker 映像 ID – 已预先填充，但可以编辑。

实例配置

- AMI 来源（必填）-确定要用作容器构建和测试实例基础映像的自定义 AMI。这可以是 AMI ID 或包含 AMI ID 的 AWS Systems Manager (SSM) 参数存储参数。

- AMI ID — 此设置未预先填入您的原始条目。输入您的基础映像的 AMI ID。示例：*ami-1234567890abcdef1*。
- SSM 参数-输入包含基础映像的 AMI ID 的 SSM 参数存储参数的名称或 ARN。示例：*/ib/test/param* 或 arn:aws:ssm:*us-east-1*:*111122223333*:parameter*/ib/test/param*。

- 存储（卷）

  EBS 第 1 卷（AMI 根）— 已预先填充。您无法编辑根卷设备名称、快照或 IOPS 选项。但是，您可以更改所有其余设置，例如大小。您还可以添加新卷。

  > **Note**
  >
  > 如果您指定了从另一个账户共享的基本 AMI，则指定的任何辅助卷的快照还必须与您的账户共享。

工作目录

- 工作目录路径 – 已预先填充，但可编辑。

组件

- 组件 - 已包含在配方中的组件显示在每个组件列表（构建和测试）末尾的选定组件部分中。您可以移除所选组件或对其重新排序，以满足您的需要。

  CIS 加固组件未遵循 Image Builder 配方中的标准组件排序规则。CIS 强化组件始终最后运行，以确保基准测试针对您的输出映像运行。

  > **Note**
  >
  > 构建和测试组件列表根据组件所有者类型显示可用组件。要添加组件，请选择添加构建组件，然后选择适用的所有权筛选器。例如，要添加与 AWS Marketplace 产品关联的构建组件，请选择AWS Marketplace。这将在控制台界面的右侧打开一个列出 AWS Marketplace 组件的选择面板。
  > 对于 CIS 组件，请选择Third party managed。

  您可以为自己的所选组件配置以下设置：

- 版本控制选项 - 已预先选择，但您可以对其进行更改。我们建议您选择使用最新的可用组件版本选项，以确保您的映像版本始终使用最新版本的组件。如果您需要在配方中使用特定的组件版本，则可以选择指定组件版本，然后在出现的组件版本框中输入版本。
- 输入参数 - 显示组件接受的输入参数。该值预先填充了配方先前版本中的值。如果您在此配方中首次使用此组件，并且已为该输入参数定义了默认值，则默认值以灰色文本显示在值框中。如果未输入其他值，Image Builder 将使用默认值。

  如果需要输入参数，但组件中未定义默认值，则必须提供一个值。如果缺少任何必需的参数且未定义默认值，Image Builder 将不会创建配方版本。

  > **Important**
  >
  > 组件参数是纯文本值，并且已记录在 AWS CloudTrail中。我们建议您使用 AWS Secrets Manager 或 P AWS Systems Manager arameter Store 来存储您的密钥。有关 Secrets Manager 的更多信息，请参阅AWS Secrets Manager 用户指南中的什么是 Secrets Manager?。有关 AWS Systems Manager Parameter Store 的更多信息，请参阅《AWS Systems Manager 用户指南》中的AWS Systems Manager Parameter Store。

要展开版本控制选项或输入参数的设置，可以选择设置名称旁边的箭头。要展开所有选定组件的所有设置，可以关闭和打开全部展开开关。

Dockerfile 模板

- Dockerfile 模板 – 已预先填充，但可编辑。您可以指定以下任意上下文变量，Image Builder 在运行时将这些变量替换为构建信息。

  parentImage（必需）

  在构建时，此变量会解析为配方的基础映像。

  示例：

  ```
  FROM
  {{{ imagebuilder:parentImage }}}
  ```

environments（如果指定了组件，则为必需）

此变量将解析为运行组件的脚本。

示例：

```
{{{ imagebuilder:environments }}}
```

components（可选）

Image Builder 解析容器配方中包含的组件的构建和测试组件脚本。此变量可以置于 Dockerfile 中的任意位置，在 environments 变量之后。

示例：

```
{{{ imagebuilder:components }}}
```

目标存储库

- 目标存储库名称 – 如果您的管道的分发配置中没有为管道运行区域（区域 1）指定其他存储库，则输出映像存储在 Amazon ECR 存储库。

创建新的容器配方版本：

1. 在容器配方详细信息页面的顶部，选择Create new version（创建新版本）。您将进入容器配方的创建配方页面。
2. 要创建新版本，请进行更改，然后选择Create recipe（创建配方）。

有关如何在创建映像管道时创建容器配方的更多信息，请参阅本指南的入门章节中的步骤 2：选择配方。

使用创建容器配方 AWS CLI

要使用中的`imagebuilder create-container-recipe`命令创建 Image Builder 容器配方 AWS CLI，请执行以下步骤：

先决条件

在运行本节中的 Image Builder 命令使用创建容器配方之前 AWS CLI，必须创建配方将使用的组件。以下步骤中的容器配方示例引用了在本指南 从中创建自定义组件 AWS CLI 章节中创建的示例组件。

创建组件后，或者如果您使用的是现有组件，请记下要ARNs 包含在配方中的组件。

1. 创建 CLI 输入 JSON 文件

   您可以使用内联命令参数为 create-container-recipe 命令提供所有输入。但是，生成的命令可能会很长。为了简化命令，您可以改为提供一个包含所有容器配方设置的 JSON 文件

   > **Note**
   >
   > JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 操作请求参数，请参阅《EC2 Image Builder API 参考》中的CreateContainerRecipe命令。
   > 要将数据值作为命令行参数提供，请参阅《AWS CLI 命令引用》中指定的参数名称。

   以下是此示例中参数的摘要：

   - 组件（对象数组，必填）- 包含 `ComponentConfiguration` 对象数组。必须指定至少一个构建组件：

     > **Note**
     >
     > Image Builder 按照您在配方中指定的顺序安装组件。但是，CIS 强化组件始终最后运行，以确保基准测试针对您的输出映像运行。

     - componentARN（字符串，必填）– 组件 ARN。

       > **Tip**
       >
       > 要使用该示例来创建自己的容器配方，请将该 ARNs 示例 ARNs 替换为用于配方的组件。其中包括每个版本的 AWS 区域、名称和版本号。

     - 参数（对象数组）- 包含 `ComponentParameter` 对象数组。如果需要输入参数，但组件中未定义默认值，则必须提供一个值。如果缺少任何必需的参数且未定义默认值，Image Builder 将不会创建配方版本。

> **Important**
>
> 组件参数是纯文本值，并且已记录在 AWS CloudTrail中。我们建议您使用 AWS Secrets Manager 或 P AWS Systems Manager arameter Store 来存储您的密钥。有关 Secrets Manager 的更多信息，请参阅AWS Secrets Manager 用户指南中的什么是 Secrets Manager?。有关 AWS Systems Manager Parameter Store 的更多信息，请参阅《AWS Systems Manager 用户指南》中的AWS Systems Manager Parameter Store。

  - 名称（字符串，必填）- 要设置的组件参数的名称。
  - 值（字符串数组，必填）- 包含用于设置指定组件参数值的字符串数组。如果为组件定义了默认值，但未提供其他值，则 AWSTOE 使用默认值。
- containerType（字符串，必填）- 要创建的容器的类型。有效值包括 DOCKER。
- dockerfileTemplateData（字符串）— 用于构建映像的 Dockerfile 模板，表示为内联数据 blob。
- 名称（字符串，必填）- 容器配方的名称。
- 描述（字符串）- 容器配方的描述。
- p@@ arentImag e（字符串，必填）— 要在容器配方中用作自定义映像基线的 Docker 容器镜像。
  - 公共图片托管在 DockerHub
  - 亚马逊 ECR 中的现有容器镜像
  - 亚马逊管理的容器镜像
- platformOverride（字符串）– 指定使用自定义基础映像时的操作系统平台。
- semanticVersion（字符串，必填）– 容器配方的语义版本按以下格式指定，每个位置都有数值以表示特定版本：<major>.<minor>.<patch>。例如，1.0.0 就是一个示例。要了解有关 Image Builder 资源的语义版本控制的更多信息，请参阅Image Builder 中的语义版本控制。
- 标签（字符串映射）– 附加到容器配方的标签。
- instanceConfiguration（对象）– 可用于配置实例以构建和测试容器映像的一组选项。
  - imag e（字符串）— 容器构建和测试实例的基础镜像。它可以包含 AMI ID，也可以指定 AWS Systems Manager (SSM) 参数存储参数，前缀为参数名称或 ARNssm:，其前缀为参数名称或 ARN。如果您使用 SSM 参数，则该参数值必须包含 AMI ID。如果您未指定基础映像，Image Builder 会使用相应的 Amazon ECS 优化的 AMI 作为基础映像。

- blockDeviceMappings（对象数组）— 定义要连接的块设备，以便从image参数中指定的 Image Builder AMI 构建实例。
  - deviceName（字符串）- 这些映射适用的设备。
  - ebs（对象）– 用于管理此映射的特定于 Amazon EBS 的配置。
    - deleteOnTermination（布尔值）-用于配置关联设备终止时的删除。
    - 已加密（布尔值）- 用于配置设备加密。
    - volumeSize（整数）- 用于覆盖设备的卷大小。
    - volumeType（字符串）- 用于覆盖设备的卷类型。
- targetRepository（对象，必填）– 如果管道的分发配置中没有为管道运行区域（区域 1）指定其他存储库，则为容器映像的目标存储库。
  - repositoryName（字符串，必填）– 存储输出容器映像的容器存储库的名称。此名称以存储库位置作为前缀。
  - 服务（字符串，必填）– 指定注册此映像的服务。
- workingDirectory（字符串）– 构建和测试工作流期间使用的工作目录。

```
{
 "components": [
   {
   "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/
helloworldal2/x.x.x"
   }
 ],
 "containerType": "DOCKER",
 "description": "My Linux Docker container image",
 "dockerfileTemplateData": "FROM
 {{{ imagebuilder:parentImage }}}\n{{{ imagebuilder:environments }}}\n{{{ imagebuilder:comp
 "name": "amazonlinux-container-recipe",
 "parentImage": "amazonlinux:latest",
 "platformOverride": "Linux",
 "semanticVersion": "1.0.2",
 "tags": {
   "sometag" : "Tag detail"
 },
 "instanceConfiguration": {
   "image": "ami-1234567890abcdef1",
   "blockDeviceMappings": [
   {
```

```
    "deviceName": "/dev/xvda",
    "ebs": {
     "deleteOnTermination": true,
     "encrypted": false,
     "volumeSize": 8,
     "volumeType": "gp2"
     }
     }
    ]
 },
 "targetRepository": {
    "repositoryName": "myrepo",
    "service": "ECR"
 },
 "workingDirectory": "/tmp"
}
```

2. 创建配方

使用以下命令以创建配方。在 `--cli-input-json` 参数中提供您在上一步中创建的 JSON 文件的名称：

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

清理资源

为避免意外费用，请务必清理根据本指南中的示例创建的资源和管道。有关删除 Image Builder 中的资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。

管理 Image Builder 基础设施配置

您可以使用基础设施配置，以指定 Image Builder 用于构建和测试 EC2 Image Builder 映像的 Amazon EC2 基础设施。基础设施设置包括：

- 构建和测试基础设施的实例类型。我们建议指定多种实例类型，因为这允许 Image Builder 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。

  对于 Mac 映像，可能需要选择原生支持 macOS 操作系统的实例类型。有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 Mac 实例。
- 实例放置设置，您可以在其中指定从映像启动的实例应保存到的主机、主机置放群组或可用区。
- 实例配置文件为您的构建和测试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 Image Builder 成功与实例进行通信的最低权限。有关更多信息，请参阅 准备好使用 Image Builder 构建自定义映像。
- 管道构建和测试实例的 VPC、子网和安全组。
- Image Builder 存储构建和测试应用程序日志的 Amazon S3 位置。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。
- 在您的构建失败并且您将 `terminateInstanceOnFailure` 设置为 `false` 时，Amazon EC2 密钥对可以让您登录到实例以进行故障排除。
- Image Builder 发送事件通知的 SNS 主题。有关 Image Builder 如何与 Amazon SNS 集成的更多信息，请参阅 Image Builder 中的 Amazon SNS 集成。

> **Note**
>
> 如果 SNS 主题已加密，则加密此主题的密钥必须位于 Image Builder 服务运行的账户中。Image Builder 无法向使用其他账户密钥进行加密的 SNS 主题发送通知。

您可以使用 Image Builder 控制台，通过 Image Builder API 或 AWS CLI中的 imagebuilder 命令来创建和管理基础设施配置。

内容

- 更新基础设施配置
- Image Builder 和 AWS PrivateLink 接口 VPC 终端节点

> **Tip**
>
> 在您具有相同类型的许多资源时，制作标签可帮助您根据分配给特定资源的标签来识别它。有关使用 Image Builder 命令为资源添加标签的更多信息 AWS CLI，请参阅本指南的标记资源部分。

列出并查看有关基础设施配置的详细信息

本部分描述您可以通过多种方式查找有关您的 EC2 Image Builder 基础设施配置的信息和查看详细信息。

基础设施配置的详细信息

- 从中列出基础架构配置 AWS CLI
- 从中获取基础架构配置的详细信息 AWS CLI

从中列出基础架构配置 AWS CLI

以下示例说明了如何使用 AWS CLI中的 list-infrastructure-configurations 列出所有基础设施配置。

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

从中获取基础架构配置的详细信息 AWS CLI

以下示例说明如何使用中的get-infrastructure-configuration命令通过指定基础设施配置的 Amazon 资源名称 (ARN) 来获取基础设施配置的详细信息。 AWS CLI

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

创建基础设施配置

本节介绍如何使用 Image Builder 控制台或中的imagebuilder命令 AWS CLI 来创建基础架构配置，

Console

要从 Image Builder 控制台创建基础设施配置资源，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中选择基础设施配置。
3. 选择创建基础设施配置。
4. 在 概述 部分中，输入以下必要信息：
   - 输入基础设施配置资源的名称。
   - 选择希望与构建和测试实例上的组件权限实例配置文件关联的 IAM 角色。Image Builder 使用这些权限来下载和运行您的组件 CloudWatch、向其上传日志，以及执行配方中组件指定的任何其他操作。
5. 在 AWS 基础设施面板中，您可以配置其余可用的基础设施设置。输入以下必要信息：
   - 实例类型：您可以为此构建指定一个或多个实例类型。该服务将根据可用性选择其中一种实例类型。

     > **Note**
     >
     > Mac 实例运行于专属主机上的 `.metal` 实例类型。您的实例类型必须与为运行它的主机定义的类型之一相匹配。有关 Mac 实例的更多信息以及原生支持 macOS 操作系统的实例类型列表，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 Mac 实例。

   - SNS 主题（可选）- 选择一个 SNS 主题来接收来自 EC2 Image Builder 的通知和提醒。

   如果您没有为以下设置提供值，则这些设置将使用特定于服务的默认值（如适用）。

   - VPC、子网和安全组：Image Builder 将使用默认 VPC 和子网。有关配置 VPC 接口端点的更多信息，请参阅 Image Builder 和 AWS PrivateLink 接口 VPC 终端节点。
   - 在故障排除设置部分，配置以下值：
     - 默认情况下，失败时终止实例复选框为选中状态。但是，构建失败时，您可以登录 EC2 实例排查问题。如果您希望实例在构建失败后继续运行，请清除该复选框。
     - 密钥对：如果 EC2 实例在构建失败后继续运行，则可以创建密钥对或使用现有密钥对登录实例并排查问题。

- 日志：您可以指定一个 S3 存储桶，Image Builder 可以在其中写入应用程序日志，以帮助构建和测试排查问题。如果您未指定 S3 存储桶，Image Builder 会将应用程序日志写入实例。

- 在实例元数据设置部分，您可以配置以下值以应用于 Image Builder 用于构建和测试映像的 EC2 实例：
  - 选择元数据版本，确定 EC2 是否要求为元数据检索请求提供签名令牌标头。
    - V1 和 V2（令牌可选）：如果您未选择任何内容，则为默认值。
    - V2（令牌必填）

    > **Note**
    >
    > 我们建议您将 Image Builder 从管道构建中启动的所有 EC2 实例配置为使用，IMDSv2 以便实例元数据检索请求需要签名的令牌标头。

  - 元数据标记响应跃点限制 – 元数据令牌可以传输的网络跃点数。最小跃点数：1，最大跃点数：64，默认为一跃点。
- 在实例放置设置部分，您可以配置以下值以应用于 Image Builder 用于构建和测试映像的 EC2 实例：
  - 您可以选择 Image Builder 在创建映像期间启动实例的可用区。
  - （可选）为运行您启动的实例的服务器选择租赁。默认情况下，EC2 实例将在共享租赁硬件上运行。这表示多个 AWS 账户 可能会共享相同物理硬件。具有 `dedicated` 租期的实例在单租户硬件上运行。具有 `host` 租期的实例在专属主机上运行。

    在构建自定义映像之前，Mac 实例需要一台作为先决条件创建的专属主机。为您的 macOS 映像选择 `host`。然后，您可以选择目标主机或主机资源组来启动实例，但如果您的专属主机启用了自动置放功能，则不需要这样做。有关更多信息，请参阅《Amazon EC2 用户指南》中的自动置放。
    - 租赁主机 ID - 运行实例的专属主机的 ID。
    - 租赁主机资源组 - 要在其中启动实例的主机资源组的 Amazon 资源名称 (ARN)。

6. 在基础设施标签部分（可选）中，您可以将元数据标签分配给 Image Builder 在构建过程中启动的 Amazon EC2 实例。输入标签作为键值对。
7. 在标签部分（可选）中，您可以将元数据标签分配给 Image Builder 作为输出创建的基础设施配置资源。输入标签作为键值对。

AWS CLI

以下步骤说明如何使用 AWS CLI中的 Image Builder create-infrastructure-configuration 命令为您的映像配置基础设施。步骤 2 中的命令采用您在步骤 1 中创建的文件。对于这些示例，步骤 1 中的文件称为 `create-infrastructure-configuration.json`。

1. 创建 CLI 输入 JSON 文件

   以下示例展示了您可能为基础设施配置创建的 JSON 文件的变体。使用文件编辑工具创建您自己的 JSON 文件。

   示例 1：用于保留编译失败后的实例的配置

   此示例指定了两种实例类型，即 `m5.large` 和 `m5.xlarge`。我们建议指定多种实例类型，因为这允许 Image Builder 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。

   `instanceProfileName` 规定了实例配置文件，该文件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 Image Builder 成功与实例进行通信的最低权限。有关更多信息，请参阅 准备好使用 Image Builder 构建自定义映像。

```
{
    "name": "ExampleInfraConfigDontTerminate",
    "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 2：具有自动置放功能的 macOS 配置

此示例为专属主机启用了自动置放功能的 Mac 实例指定实例类型和置放位置。

```
{
   "name": "macOSInfraConfigAutoPlacement",
   "description": "An example infrastructure configuration for macOS.",
   "instanceProfileName": "EC2InstanceProfileForImageBuilder",
   "instanceTypes": ["mac1.metal, mac2.metal"],
   "terminateInstanceOnFailure": false,
   "placement": {
      "tenancy": "host"
   }
}
```

示例 3：指定了主机 ID 的 macOS 配置

此示例为以特定专属主机为目标的 Mac 实例指定实例类型和置放位置。

```
{
   "name": "macOSInfraConfigHostPlacement",
   "description": "An example infrastructure configuration for macOS.",
   "instanceProfileName": "EC2InstanceProfileForImageBuilder",
   "instanceTypes": ["mac2-m1ultra.metal"],
   "terminateInstanceOnFailure": false,
   "placement": {
      "tenancy": "host",
      "hostId" : "h-1234567890abcdef0"
   }
}
```

2. 当您运行以下命令时，使用您作为输入而创建的文件。

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

更新基础设施配置

本节介绍如何使用 Image Builder 控制台或中的imagebuilder命令 AWS CLI 来更新基础设施配置资源。要跟踪资源，可以应用标签，如下所示。输入标签作为键值对。

- 资源标签将元数据标签分配给 Image Builder 在构建过程中启动的 Amazon EC2 实例。
- 标签将元数据标签分配给 Image Builder 作为输出创建的基础设施配置资源。

Console

您可以从 Image Builder 控制台编辑以下基础设施配置详细信息：

- 基础设施配置的描述。
- 将 IAM 角色 与实例配置文件关联。
- AWS 基础架构，包括实例类型和通知的 SNS 主题。
- VPC、子网和安全组。
- 问题排查设置（包括失败时终止实例、用于连接的密钥对以及用于存储实例日志的可选 S3 存储桶位置）。

要通过 Image Builder 控制台更新基础设施配置资源，请执行以下步骤：

配置选择现有的 Image Builder 基础设施配置

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 要查看您账户下的基础设施配置资源列表，请从导航窗格中选择基础设施配置。
3. 要查看详细信息或编辑基础设施配置，请选择配置名称链接。此操作将打开基础设施配置的详细视图。

> **Note**
>
> 您也可以选中配置名称旁边的框，然后选择查看详细信息。

4. 在基础设施详细信息面板的右上角，选择编辑。
5. 准备保存对基础设施配置所做的更新时，请选择保存更改。

AWS CLI

以下示例展示了如何使用 AWS CLI中的 Image Builder update-infrastructure-configuration 命令更新映像的基础设施配置。

1. 创建 CLI 输入 JSON 文件

   此基础设施配置示例使用的设置与创建示例相同，仅仅将 `terminateInstanceOnFailure` 设置更新为 `false`。运行 update-infrastructure-configuration 命令后，使用此基础设施配置的管道将在构建失败时终止构建和测试实例。

   使用文件编辑工具创建一个 JSON 文件，其中包含以下示例中显示的密钥，以及对您的环境有效的值。此示例使用名为 `update-infrastructure-configuration.json` 的文件：

   ```
   {
   "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"
   }
   ```

2. 当您运行以下命令时，使用您作为输入而创建的文件。

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

Image Builder 和 AWS PrivateLink 接口 VPC 终端节点

您可以通过创建接口 VPC 端点在 VPC 和 EC2 Image Builder 之间建立私有连接。接口端点由一项技术提供支持 AWS PrivateLink，该技术使您 APIs 无需互联网网关、NAT 设备、VPN 连接或 Direct

Connect 连接即可私密访问 Image Builder。您的 VPC 中的实例不需要公有 IP 地址即可与 Image Builder 通信 APIs。VPC 和 Image Builder 之间的流量不会脱离 Amazon 网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。创建新映像时，可以在基础设施配置中指定 VPC 子网 ID。

> **Note**
>
> 您从 VPC 内访问的每项服务都有自己的接口端点和自己的端点策略。Image Builder 下载 AWSTOE 组件管理器应用程序并访问 S3 存储桶中的托管资源以创建自定义映像。要授予访问这些存储桶的权限，您必须更新 S3 端点策略以允许访问。有关更多信息，请参阅 访问 S3 存储桶的自定义策略。

有关 VPC 端点的更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 端点 (AWS PrivateLink)。

Image Builder VPC 端点的注意事项

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

Image Builder 支持从 VPC 调用它的所有 API 操作。

为 Image Builder 创建接口 VPC 端点

要为 Image Builder 服务创建 VPC 终端节点，您可以使用亚马逊 VPC 控制台或 AWS Command Line Interface (AWS CLI)。有关更多信息，请参阅《Amazon VPC User Guide》中的 Creating an interface endpoint。

使用以下服务名称为 Image Builder 创建 VPC 端点：

- com.amazonaws。 *region*.imagebuilder

如果为终端节点启用私有 DNS，则可以使用其默认 DNS 名称作为区域，向 Image Builder 发送 API 请求，例如：`imagebuilder.us-east-1.amazonaws.com`。要查找适用于您的目标区域的端点，请参阅 Amazon Web Services 一般参考 中的 EC2 Image Builder 端点和配额。

有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

为 Image Builder 创建 VPC 端点策略

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

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

如果您在配方中使用 Amazon 托管的组件，则 Image Builder 的 VPC 端点必须允许访问以下服务拥有的组件库：

`arn:aws:imagebuilder:region:aws:component/*`

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

有关更多信息，请参阅《Amazon VPC User Guide》中的 Controlling access to services with VPC endpoints。

访问 S3 存储桶的自定义策略

Image Builder 使用公开可用的 S3 存储桶来存储和访问托管的资源，例如组件。它还会从单独的 S3 存储桶下载 AWSTOE 组件管理应用程序。如果您在环境中使用 Amazon S3 的 VPC 端点，则需要确保您的 S3 VPC 端点策略允许 Image Builder 访问以下 S3 存储桶。每个 AWS 区域 (*region*) 和应用程序环境 (*environment*) 的存储桶名称都是唯一的。Image Builder 并 AWSTOE 支持以下应用程序环境：`prod`、`preprod`、和`beta`。

- AWSTOE 组件管理器存储桶：

  ```
  s3://ec2imagebuilder-toe-region-environment
  ```

  示例：s3://ec2 imagebuilder-toe-us-west -2-prod/*

- Image Builder 托管资源存储桶：

```
s3://ec2imagebuilder-managed-resources-region-environment/components
```

示例：s3://ec2 imagebuilder-managed-resources-us-west-2-prod/components/*

VPC 端点策略示例

本节包括自定义 VPC 端点策略的示例。

Image Builder 操作的常规 VPC 端点策略

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

JSON

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

按组织限制访问权限，允许访问托管组件

以下端点策略示例说明了如何对属于您组织的身份和资源的访问权限进行限制，以及如何提供对 Amazon 托管的 Image Builder 组件的访问权限。将*regionprincipal-org-id*、和，替换*resource-org-id*为贵组织的价值观。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "principal-org-id",
                    "aws:ResourceOrgID": "resource-org-id"
                }
            }
        },
        {
            "Sid": "AllowAccessToEC2ImageBuilderComponents",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "imagebuilder:GetComponent"
            ],
            "Resource": [
                "arn:aws:imagebuilder:us-east-1:aws:component/*"
            ]
        }
    ]
}
```

访问 Amazon S3 存储桶的 VPC 端点策略

以下 S3 端点策略示例显示了如何提供对 Image Builder 用于构建自定义映像的 S3 存储桶的访问权限。*environment*用贵组织的价值观替换*region*和。根据您的应用程序要求向策略添加任何其他必需的权限。

> **Note**
>
> 对于 Linux 映像，如果您未在映像配方中指定用户数据，Image Builder 会添加一个脚本，用于在映像的构建和测试实例上下载和安装 Systems Manager Agent。要下载代理，Image Builder 会访问您的构建区域的 S3 存储桶。
>
> 为确保 Image Builder 可以引导构建和测试实例，请在 S3 端点策略中添加以下额外资源：`"arn:aws:s3:::amazon-ssm-region/*"`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowImageBuilderAccessToAppAndComponentBuckets",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::ec2imagebuilder-toe-region-environment/*",
        "arn:aws:s3:::ec2imagebuilder-managed-resources-region-environment/
components/*"
      ]
    }
  ]
}
```

管理 Image Builder 分配设置

在为输出映像配置分配设置之前，我们建议您针对在分配目标区域中从您的输出映像启动的实例，验证所有底层基础设施的可用性或其他要求。例如，并非所有区域都支持 EC2 Mac 专属主机，而这些主机对于从 macOS 映像启动实例是必需的。有关专属主机的实例类型和定价的更多信息，请参阅 Amazon EC2 专属主机定价。

使用 Image Builder 创建分配设置后，您可以使用 Image Builder 控制台、Image Builder API 或 AWS CLI中的 imagebuilder 命令对其进行管理。通过使用分配设置，您可以执行以下操作：

AMI 分配

- 指定输出 AMI 的名称和描述。
- 授权其他组织 AWS 账户、组织以及 OUs 从所有者的账户启动 AMI。所有者账户需要支付与 AMI 相关的费用。

> **Note**
>
> 要公开 AMI，请将启动许可授权账户设置为 `all`。有关相关信息和示例，请参阅《Amazon EC2 API 参考》中的 ModifyImageAttribute。

- 为每个指定的目标账户、组织和目标区域创建输出 AMI OUs 的副本。目标账户、组织和 OUs 拥有自己的 AMI 副本，并需支付任何相关费用。有关向 AWS Organizations 和分发 AMI 的更多信息 OUs，请参阅与组织共享 AMI 或 OUs。
- 将 AMI 复制到其他所有者的账户 AWS 区域。
- 将 VM 映像磁盘导出到 Amazon Simple Storage Service (Amazon S3)。有关更多信息，请参阅 示例：为来自的输出 VM 磁盘创建分发设置 AWS CLI。

容器映像分配

- 指定 Image Builder 在分配区域中存储输出映像的 ECR 存储库。

您可以通过以下方式使用分发设置，将图像一次性传送到目标区域、账户 AWS Organizations 和组织单位 (OUs)，或者每次构建管道：

- 要自动将更新的图像传送到指定的区域、账户、Organizations 和 OUs，请将分发设置与按计划运行的 Image Builder 管道配合使用。

- 要创建新映像并将其交付给指定的区域、账户、组织和，请使用 “操作” 菜单中的 “运行管道” OUs，将分发设置与 Image Builder 管道一起使用，该管道从图像生成器控制台运行一次。
- 要创建新映像并将其交付给指定的区域、账户、组织和，请使用分发设置以及 OUs以下 API 操作或 Image Builder 命令 AWS CLI：
  - Image Builder API 中的 CreateImage 操作。
  - AWS CLI中的 create-image 命令。
- 作为常规映像构建过程的一部分，将虚拟机 (VM) 映像磁盘导出到目标区域的 S3 存储桶。

> **Tip**
>
> 在您具有相同类型的许多资源时，标记可帮助您根据分配给特定资源的标签来识别它。有关使用 Image Builder 命令为资源添加标签的更多信息 AWS CLI，请参阅本指南的标记资源部分。

内容

列出和查看分配配置详细信息

本章节介绍查找 EC2 Image Builder 分配配置信息和查看详细信息的多种方法。

分配配置详细信息

通过控制台列出分配配置

要在 Image Builder 控制台中查看在您的帐户下创建的分配配置列表，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择分配设置。这将显示在您的账户下创建的分配配置的列表。
3. 要查看详细信息或创建新的分配配置，请选择配置名称链接。这将打开分配设置的详细视图。

> **Note**
>
> 您也可以选中配置名称旁边的框，然后选择查看详细信息。

通过控制台查看分配配置详细信息

要使用 Image Builder 控制台查看特定分配配置的详细信息，请使用 通过控制台列出分配配置 中描述的步骤选择要查看的配置。

在分配详情页面上，您可以：

- 删除分配配置。有关删除 Image Builder 中的资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。
- 编辑分配的详细信息。

列出来自 AWS CLI

以下示例说明如何使用中的list-distribution-configurations命令列 AWS CLI 出所有发行版。

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

从中获取分发配置的详细信息 AWS CLI

以下示例说明如何使用中的get-distribution-configuration命令通过指定分配配置的 Amazon 资源名称 (ARN) 来获取分配配置的详细信息。 AWS CLI

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

创建和更新 AMI 分配配置

本节介绍如何创建和更新 Image Builder AMI 的分配配置。

内容

AMI 分发先决条件

某些分发设置有先决条件，如下所示：

主题

SSM 输出参数的先决条件

在创建用于设置 AWS Systems Manager 参数存储参数（SSM 参数）的新 AMI 分发配置之前，请确保满足以下先决条件。

执行角色

在中创建管道或使用 create-image 命令时 AWS CLI，只能指定一个 Image Builder 执行角色。如果您定义了 Image Builder 工作流程执行角色，则需要向该角色添加任何其他功能权限。否则，您将创建一个包含所需权限的新自定义角色。

- 要在分发期间将输出 AMI ID 存储在 SSM 参数中，您必须在 Image Builder 执行角色中指定`ssm:PutParameter`操作，并将该参数列为资源。
- 当您将参数数据类型设置为`AWS EC2 Image`以向 Systems Manager 发出信号，要求其将参数值作为 AMI ID 进行验证时，还必须添加`ec2:DescribeImages`操作。

EC2 快速启动的先决条件

在为适用于 Windows AMIs 的 EC2 Fast Launch 创建新的分配配置之前，请确保满足以下先决条件。

- 如果在配置 EC2 Fast Launch 时提供了自定义启动模板，则该服务将使用您在启动模板中定义的 VPC 和其他配置设置。有关更多信息，请参阅在设置 EC2 快速启动时使用启动模板。
- 如果您不使用自定义启动模板来配置设置，则必须将EC2FastLaunchFullAccess策略附加到 Image Builder 用于创建映像的 IAM 角色。在中创建管道或使用 create-image 命令时 AWS CLI，只能指定一个 Image Builder 执行角色。如果您定义了 Image Builder 工作流程执行角色，则需要向该角色添加任何其他功能权限。否则，您将创建一个包含所需权限的新自定义角色。

  然后，当 Image Builder 复制您的映像时，EC2 Fast Launch 会自动创建一个 CloudFormation 堆栈，其中包含以下资源 AWS 账户。

  - 虚拟私有云（VPC）
  - 跨多个可用区的若干私有子网
  - 使用实例元数据服务版本 2 配置的启动模板 (IMDSv2)
  - 一个没有任何入站或出站规则的安全组

> **Note**
>
> 对于预先启用 EC2 Fast Launch 的 AMI，Image Builder 不支持跨账户分配。必须从目标账户启用 EC2 Fast Launch。

创建 AMI 分发配置

分发配置包括输出 AMI 名称、用于加密的特定区域设置、启动权限以及可以启动输出 AMI 的组织和组织单位 (OUs) 以及许可证配置。 AWS 账户

分布配置允许您指定输出 AMI 的名称和描述，授权其他 AWS 账户 人启动 AMI，将 AMI 复制到其他账户，以及将 AMI 复制到其他 AWS 区域。它还允许你将 AMI 导出到亚马逊简单存储服务 (Amazon S3)，或者为输出 Windows 配置 EC2 快速启动。AMIs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。

Console

按照以下步骤在中创建新的 AMI 分发配置 AWS 管理控制台：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择分配设置。这将显示在您的账户下创建的分配配置的列表。
3. 选择分配设置面板顶部附近的创建分配设置。
4. 在映像类型部分，选择 Amazon Machine Image (AMI) 输出类型。
5. 在常规部分，输入分配配置的名称和可选描述。
6. 在区域设置部分，输入您要分配 AMI 的每个区域的以下详细信息：

   a. 默认情况下，AMI 分配到当前区域（区域 1）。区域 1 是分配的来源。区域 1 的某些设置未开放编辑功能。对于您添加的任何区域，您可以从区域下拉列表中选择一个区域。

   Kms 密钥标 AWS KMS key 识用于加密目标区域中映像的 EBS 卷。请务必注意，这不适用于该构建在源区域（区域 1）中使用您的账户创建的原始 AMI。在该构建的分配阶段运行的加密仅适用于分配给其他账户或地区的映像。

   要对在源区域为您的账户创建的 AMI 的 EBS 卷进行加密，您必须在映像配方块设备映射（控制台中的存储（卷））中设置 KMS 密钥。

   Image Builder 将 AMI 复制到您为该区域指定的目标账户。

   > **先决条件**
   >
   > 要跨账户复制映像，您必须在所有分配目标账户中创建 `EC2ImageBuilderDistributionCrossAccountRole` 角色，并将 Ec2ImageBuilderCrossAccountDistributionAccess 策略 托管策略附加到该角色。

   输出 AMI 名称为可选项。如果您提供了名称，则最终的输出 AMI 名称会附加一个 AMI 构建时间的时间戳。如果未指定名称，Image Builder 会将构建时间戳附加到配方名称。这样可以确保每次构建的 AMI 名称都是唯一的。

   i. 通过 AMI 共享，您可以向指定的 AWS 委托人授予从您的 AMI 启动实例的访问权限。如果展开 AMI 共享部分，则可以输入以下详细信息：

      - 启动权限 — 如果您想保持 AMI 私有，请选择私有，并允许特定 AWS 委托人访问您的私有 AMI 启动实例。如果要使 AMI 公开，请选择公开。任何 AWS 委托人都可以从您的公有 AMI 启动实例。
      - 委托人-您可以为以下类型的 AWS 委托人授予启动实例的访问权限：

- AWS 账户-授予对特定 AWS 账户的访问权限
- 组织单位 (OU) — 授予对 OU 及其所有子实体的访问权限。子实体包括 OUs 和 AWS 帐户。
- 组织-授予您 AWS Organizations及其所有子实体的访问权限。子实体包括 OUs 和 AWS 帐户。

首先，选择主体类型。然后在下拉列表右侧的框中，输入要授予访问权限的 AWS 主体的 ID。您可以输入 IDs 多种不同的类型。

ii. 您可以展开 “许可配置” 部分，将使用创建的许可配置附加 AWS License Manager 到 Image Builder 映像。许可证配置包含基于您的企业协议条款的许可规则。Image Builder 会自动包含与您的基本 AMI 关联的许可证配置。

iii. 您可以展开启动模板配置部分，指定用于从您创建的 AMI 启动实例的 EC2 启动模板。

如果您使用的是 EC2 启动模板，则可以指示 Image Builder 在构建完成后创建包含最新 AMI ID 的启动模板的新版本。要更新启动模板，请按以下方式配置设置：

- 启动模板名称 — 选择您希望 Image Builder 更新的启动模板的名称。
- 设置默认版本 — 选中此复选框可将启动模板的默认版本更新为新版本。

要添加另一个启动模板配置，请选择添加启动模板配置。您在每个区域最多可以拥有五种启动模板配置。

iv. 您可以展开 SSM 参数配置部分来配置 SSM 参数，该参数将存储分发到目标区域的图像的输出 AMI ID。您可以选择在该地区指定一个分销账户。

参数名称-输入参数的名称。例如 `/output/image/param`。

数据类型-保留默认值 (`AWS EC2 Image`)。这会让 Systems Manager 验证参数值以确保它是有效的 AMI ID。

b. 要为其他区域添加分配设置，请选择添加区域。

7. 完成后，选择创建设置。

AWS CLI

以下示例说明如何采用 AWS CLI使用 create-distribution-configuration 命令为 AMI 创建新的分配配置。

1. 创建 CLI 输入 JSON 文件

   使用文件编辑工具创建一个 JSON 文件，其中包含以下示例之一中显示的密钥，以及对您的环境有效的值。这些示例定义了哪些 AWS 账户 AWS Organizations 或哪些组织单位 (OUs) 有权启动您分发到指定区域的 AMI。对文件 `create-ami-distribution-configuration.json` 进行命名，以便在下一步中使用：

   示例 1：分发到 AWS 账户

   此示例将 AMI 分配到两个区域，并指定在每个区域都拥有启动权限的 AWS 账户 。

```
{
 "name": "MyExampleAccountDistribution",
 "description": "Copies AMI to eu-west-1, and specifies accounts that can launch
 instances in each Region.",
 "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"
     ]
    }
```

```
    }
   }
  ]
}
```

示例 2：分发给 Organiations 和 OUs

此示例将 AMI 分配到源区域，并指定组织和 OU 启动权限。

```
{
 "name": "MyExampleAWSOrganizationDistribution",
 "description": "Shares AMI with the Organization and OU",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "name": "Name {{ imagebuilder:buildDate }}",
    "launchPermission": {
     "organizationArns": [
      "arn:aws:organizations::123456789012:organization/o-myorganization123"
     ],
     "organizationalUnitArns": [
      "arn:aws:organizations::123456789012:ou/o-123example/ou-1234-
myorganizationalunit"
     ]
    }
   }
  }
 ]
}
```

示例 3：将输出的 AMI ID 存储在 SSM 参数中

此示例将输出 AMI ID 存储在分布区域的 AWS Systems Manager 参数存储参数中。

```
{
 "name": "SSMParameterOutputAMI",
 "description": "Updates an SSM parameter with the output AMI ID for the
 distribution.",
 "distributions": [
  {
   "region": "us-west-2",
```

```
    "amiDistributionConfiguration": {
     "name": "Name {{ imagebuilder:buildDate }}"
    },
    "ssmParameterConfigurations": [
     {
      "amiAccountId": "111122223333",
      "parameterName": "/output/image/param",
      "dataType": "aws:ec2:image"
     }
    ]
   }
  ]
}
```

2. 使用创建的文件作为输入，运行以下命令。

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

有关更多详细信息，请参阅 AWS CLI 命令参考中的 create-distribution-configuration。

更新 AMI 分发配置

您可以更改 AMI 分发配置。但是，所做的更改不适用于 Image Builder 已经分配的任何资源。例如，如果您已将 AMI 分配到某个区域，但随后又将其从分配中移除，则在您手动将其移除之前，已分配的 AMI 将保留在该区域中。

AWS 管理控制台

按照以下步骤进行 AMI 分发配置 AWS 管理控制台：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。

2. 从导航窗格中，选择分配设置。这将显示在您的账户下创建的分配配置的列表。
3. 要查看详细信息或更新分配配置，请选择配置名称链接。这将打开分配设置的详细视图。

> **Note**
>
> 您也可以选中配置名称旁边的框，然后选择查看详细信息。

4. 要编辑分配配置，请从分配详细信息部分的右上角选择编辑。某些字段已锁定，例如分配配置的名称和显示为区域 1 的默认区域。有关分配配置设置的更多信息，请参阅 创建 AMI 分发配置。
5. 完成后，选择 Save changes（保存更改）。

AWS CLI

以下示例说明如何通过 AWS CLI使用 update-distribution-configuration 命令为您的 AMI 更新分配配置。

1. 创建 CLI 输入 JSON 文件

   使用文件编辑工具创建 JSON 文件，其中包含以下示例中显示的密钥以及对您的环境有效的值。此示例使用名为 `update-ami-distribution-configuration.json` 的文件。

```
{
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/update-ami-distribution-
configuration.json",
 "description": "Copies AMI to eu-west-2, and specifies accounts that can launch
 instances in each Region.",
 "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"
     ]
    }
   }
  }
 ]
}
```

2. 使用创建的文件作为输入，运行以下命令。

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

有关更多详细信息，请参阅 AWS CLI 命令参考中的 update-distribution-configuration。要更新分配配置资源的标签，请参阅 标记资源 一节。

示例：使用输出启动模板启用 EC2 快速启动 AMIs

以下示例说明如何使用带有启动模板的create-distribution-configuration命令来创建分配设置，这些设置已为您的 AMI 配置了 EC2 快速启动，请从中获得 AWS CLI。

要在没有启动模板的情况下配置 EC2 Fast Launch 设置，请确保EC2 Fast Launch 先决条件在创建分配配置之前满足所有要求。

1. 创建 CLI 输入 JSON 文件

   使用文件编辑工具创建一个 JSON 文件，其中包含以下示例中显示的密钥，以及对您的环境有效的值。

   此示例同时启动其所有目标资源的实例，因为并行启动的最大数量大于目标资源数量。该文件在下一步所示的命令示例中被命名为 `ami-dist-config-win-fast-launch.json`。

```
{
"name": "WinFastLaunchDistribution",
"description": "An example of Windows AMI EC2 Fast Launch settings in the
 distribution configuration.",
"distributions": [
 {
  "region": "us-west-2",
  "amiDistributionConfiguration": {
   "name": "Name {{imagebuilder:buildDate}}",
   "description": "Includes Windows AMI EC2 Fast Launch settings.",
   "amiTags": {
    "KeyName": "Some Value"
   }
  },
  "fastLaunchConfigurations": [{
   "enabled": true,
   "snapshotConfiguration": {
    "targetResourceCount": 5
   },
   "maxParallelLaunches": 6,
   "launchTemplate": {
    "launchTemplateId": "lt-0ab1234c56d789012",
    "launchTemplateVersion": "1"
    }
  }],
  "launchTemplateConfigurations": [{
       "launchTemplateId": "lt-0ab1234c56d789012",
       "setDefaultVersion": true
    }]
 }]
}
```

> **Note**
>
> 您可以指定 `launchTemplateName` 而非 `launchTemplate` 部分中的 `launchTemplateId`，但不能同时指定名称和 ID。

2. 使用创建的文件作为输入，运行以下命令。

```
aws imagebuilder create-distribution-configuration --cli-input-json file://ami-dist-config-win-fast-launch.json
```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

有关更多详细信息，请参阅 AWS CLI 命令参考中的 create-distribution-configuration。

示例：为来自的输出 VM 磁盘创建分发设置 AWS CLI

以下示例说明如何使用 create-distribution-configuration 命令创建分配设置，以便在每次构建映像时将 VM 映像磁盘导出到 Amazon S3。

1. 创建 CLI 输入 JSON 文件

   您可以简化在 AWS CLI中使用的 create-distribution-configuration 命令。为此，请创建一个 JSON 文件，其中包含要传递到命令中的所有导出配置。

   > **Note**
   >
   > JSON 文件中数据值的命名约定遵循为 Image Builder API 操作请求参数指定的模式。要查看 API 操作请求参数，请参阅《EC2 Image Builder API 参考》中的CreateDistributionConfiguration命令。
   > 要将数据值作为命令行参数提供，请参阅 AWS CLI 命令参考中指定的参数名，将 create-distribution-configuration 命令作为选项。

以下是我们在本示例的 s3ExportConfiguration JSON 对象中指定的参数摘要：

- RoleName（字符串，必填）-向虚拟机授予将图像导出到 S3 存储桶的 Import/Export 权限的角色名称。
- diskImageFormat（字符串，必填）-将更新的磁盘映像导出为以下支持的格式之一：
  - 虚拟硬盘 (VHD)— 可与 Citrix Xen 和 Microsoft Hyper-V 虚拟化产品兼容。
  - 流优化的 ESX 虚拟机磁盘 (VMDK) — 兼容 ESX VMware 和 VMware vSphere 版本 4、5 和 6。
  - 原始 — 原始格式。
- s3Bucket（字符串，必填）— 用于存储 VM 的输出磁盘映像的 S3 存储桶。

将该文件保存为 export-vm-disks.json。在 create-distribution-configuration 命令中使用文件名。

```
{
 "name": "example-distribution-configuration-with-vm-export",
 "description": "example",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "description": "example-with-vm-export"

   },
   "s3ExportConfiguration": {
    "roleName": "vmimport",
    "diskImageFormat": "RAW",
    "s3Bucket": "vm-bucket-export"
   }
  }],
 "clientToken": "abc123def4567ab"
}
```

2. 使用创建的文件作为输入，运行以下命令。

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

有关更多详细信息，请参阅 AWS CLI 命令参考中的 create-distribution-configuration。

创建和更新容器映像的分配设置

本章节介绍如何创建和更新 Image Builder 容器映像的分配设置。

内容

从 Image Builder 容器映像创建分发设置 AWS CLI

分发配置使您可以指定输出容器映像的名称和描述，并将容器映像复制到其他 AWS 区域。您也可以对分配配置资源和每个区域内的容器映像应用单独的标签。

1. 创建 CLI 输入 JSON 文件

   使用您常用的文件编辑工具创建一个 JSON 文件，其中包含以下示例中显示的密钥，以及对您的环境有效的值。此示例使用名为 `create-container-distribution-configuration.json` 的文件：

```
{
 "name": "distribution-configuration-name",
 "description": "Distributes container image to Amazon ECR repository in two
 regions.",
 "distributions": [
  {
   "region": "us-west-2",
   "containerDistributionConfiguration": {
    "description": "My test image.",
```

```
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
    "containerTags": ["west2", "image1"]
   }
  },
  {
   "region": "us-east-1",
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
      "containerTags": ["east1", "imagedist"]
   }
  }
 ],
 "tags": {
    "DistributionConfigurationTestTagKey1":
 "DistributionConfigurationTestTagValue1",
    "DistributionConfigurationTestTagKey2":
 "DistributionConfigurationTestTagValue2"
 }
}
```

2. 使用创建的文件作为输入，运行以下命令。

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

有关更多详细信息，请参阅 AWS CLI 命令参考中的 create-distribution-configuration。

从中更新容器映像的分发设置 AWS CLI

以下示例说明如何采用 AWS CLI使用 update-distribution-configuration 命令为容器映像创建新的分配配置。您还可以在每个区域内更新容器映像的标签。

1. 创建 CLI 输入 JSON 文件

   使用您常用的文件编辑工具创建一个 JSON 文件，其中包含以下示例所示的密钥以及对您的环境有效的值。此示例使用名为 `update-container-distribution-configuration.json` 的文件：

```
{
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/update-container-distribution-
configuration.json",
 "description": "Distributes container image to Amazon ECR repository in two
 regions.",
 "distributions": [
  {
   "region": "us-west-2",
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
    "containerTags": ["west2", "image1"]
   }
  },
  {
   "region": "us-east-2",
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
      "containerTags": ["east2", "imagedist"]
   }
  }
 ]
}
```

2. 使用创建的文件作为输入，运行以下命令：

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

有关更多详细信息，请参阅 AWS CLI 命令参考中的 update-distribution-configuration。要更新分配配置资源的标签，请参阅 标记资源 一节。

使用 Image Builder 设置跨账户 AMI 分配

本部分介绍了如何配置分配设置，以将 Image Builder AMI 传递给您指定的其他账户。

然后，目标账户可以根据需要启动或修改 AMI。

> **Note**
>
> AWS CLI 本节中的命令示例假设您之前创建了图像配方和基础设施配置 JSON 文件。要为映像配方创建 JSON 文件，请参阅 使用创建图像配方 AWS CLI。要为基础设施配置创建 JSON 文件，请参阅 创建基础设施配置。

跨账户 AMI 分配的先决条件

为确保目标账户能够成功地从您的 Image Builder 映像启动实例，您必须为所有区域的所有目标账户配置相应权限。

如果您使用 AWS Key Management Service (AWS KMS) 加密您的 AMI，则必须 AWS KMS key 为您的账户配置一个用于加密新映像的。

当 Image Builder 执行跨账户分发加密操作时 AMIs，源账户中的图像将被解密并推送到目标区域，然后使用该区域的指定密钥对其进行重新加密。由于 Image Builder 代表目标账户行事，并使用您在目标区域创建的 IAM 角色，因此该账户必须有权访问源区域和目标区域中的密钥。

加密密钥

如果您的映像使用 AWS KMS加密，则需要满足以下先决条件。IAM 先决条件将在下一部分中介绍。

源账户要求

- 在您构建和分配 AMI 的所有区域的账户中创建 KMS 密钥。您也可以使用现有密钥。
- 更新所有这些密钥的密钥策略，以允许目标账户使用您的密钥。

目标账户要求

- 向 `EC2ImageBuilderDistributionCrossAccountRole` 添加内联策略，允许该角色执行分配加密 AMI 所需的操作。有关 IAM 配置步骤，请参阅 IAM 策略 先决条件部分。

有关使用跨账户访问的更多信息 AWS KMS，请参阅AWS Key Management Service 开发者指南中的允许其他账户中的用户使用 KMS 密钥。

在映像配方中指定您的加密密钥，如下所示：

- 如果您使用的是 Image Builder 控制台，请从配方的存储（卷）部分的加密（KMS 别名）下拉列表中选择您的加密密钥。
- 如果您使用的是 CreateImageRecipe API 操作或中的create-image-recipe命令，请在 AWS CLI JSON 输入下方的`ebs`部分`blockDeviceMappings`中配置您的密钥。

  以下 JSON 片段显示了映像配方的加密设置。除了提供您的加密密钥外，您还必须将 `encrypted` 标志设置为 `true`。

```
{
 ...
 "blockDeviceMappings": [
 {
  "deviceName": "Example root volume",
  "ebs": {
   "deleteOnTermination": true,
   "encrypted": true,
   "iops": 100,
```

```
    "kmsKeyId": "image-owner-key-id",
    ...
   },
   ...
 }],
 ...
}
```

IAM 策略

要在 AWS Identity and Access Management (IAM) 中配置跨账户分配权限，请执行以下步骤：

1. 要使用跨账户分布 AMIs 的 Image Builder，目标账户所有者必须在其账户中创建一个名为的新 IAM 角色`EC2ImageBuilderDistributionCrossAccountRole`。
2. 他们必须将 Ec2ImageBuilderCrossAccountDistributionAccess 策略 附加到角色才能启用跨账户分配。有关托管策略的更多信息，请参阅 AWS Identity and Access Management 用户指南 中的托管策略与内联策略。
3. 确认源账户 ID 已添加到目标账户的 IAM 角色所附的信任策略中。以下示例显示了目标账户中的信任策略，该策略指定了源账户的账户 ID。

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": [{
           "Effect": "Allow",
           "Principal": {
               "AWS": "arn:aws:iam::444455556666:root"
           },
           "Action": "sts:AssumeRole"
       }]
   }
   ```

   有关信任策略的更多信息，请参阅 AWS Identity and Access Management 用户指南 中的 基于资源的策略。
4. 如果您分配的 AMI 已加密，则目标账户所有者必须在其账户的`EC2ImageBuilderDistributionCrossAccountRole` 中添加以下内联策略，以便使用您的

KMS 密钥。Principal 部分包含他们的账号。这使得 Image Builder 能够在使用每个 AWS KMS 区域的相应密钥对 AMI 进行加密和解密时代表他们采取行动。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "AllowRoleToPerformKMSOperationsOnBehalfOfTheDestinationAccount",
   "Effect": "Allow",
   "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
   ],
   "Resource": "*"
  }
 ]
}
```

有关内联策略的更多信息，请参阅 AWS Identity and Access Management 用户指南中的内联策略。

5. 如果您使用 launchTemplateConfigurations 指定 Amazon EC2 启动模板，则还必须在每个目标账户的 EC2ImageBuilderDistributionCrossAccountRole 中添加以下策略。

JSON

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

```
                "ec2:CreateLaunchTemplateVersion",
                "ec2:ModifyLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeLaunchTemplates"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:launch-template/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "EC2 Image Builder"
                }
            }
        }
    ]
}
```

6. 如果您使用 P AWS Systems Manager arameter Store 参数来存储分配账户和区域的输出 AMI 的 AMI ID，则必须在每个目标账户`EC2ImageBuilderDistributionCrossAccountRole`中将以下策略添加到您的中。

   JSON

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

```
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages"
            ],
            "Resource": "*"
        }
    ]
}
```

跨账户分配的限制

跨账户分配 Image Builder 映像时有一些限制：

- 每个目标区域的目标账户仅限 50 个并发 AMI 副本。
- 如果要将半虚拟化 (PV) 虚拟化 AMI 复制到另一个区域，则目标区域必须支持 PV 虚拟化 AMIs。有关更多信息，请参阅 Linux AMI 虚拟化类型。
- 您无法创建加密快照的未加密副本。如果您没有为 `KmsKeyId` 参数指定 AWS Key Management Service (AWS KMS) 客户托管密钥，Image Builder 将使用 Amazon Elastic Block Store (Amazon EBS) 的默认密钥。有关更多信息，请参阅 Amazon Elastic Compute Cloud 用户指南 中的 Amazon EBS 加密。

有关更多信息，请参阅 CreateDistributionConfigurationEC2 Image Builder API 参考中的。

通过控制台为 Image Builder AMI 配置跨账户分配

本节介绍如何 AMIs 使用创建和配置用于跨账户分发您的 Image Builder 的分发设置。 AWS 管理控制台配置跨账户分配需要特定的 IAM 权限。在继续操作之前，您必须完成本部分的 跨账户 AMI 分配的先决条件。

要在 Image Builder 控制台中创建分配，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。

2. 从导航窗格中，选择分配设置。这将显示在您的账户下创建的分配设置列表。
3. 在分配设置页面的顶部，选择创建分配设置。这将跳转到创建分配设置页面。
4. 在映像类型部分，选择 Amazon Machine Image (AMI) 作为输出类型。这是默认设置。
5. 在常规部分中，输入要创建的分配设置资源的名称（必填）。
6. 在区域设置部分，在选定区域的目标账户中输入您要向其分配 AMI 的 12 位账户 ID，然后按输入。这将检查格式是否正确，然后框的下方会显示您输入的账户 ID。重复该过程以添加更多帐户。

   要删除您输入的账户，请选择账户 ID 右侧显示的 X。

   输入每个区域的输出 AMI 名称。
7. 继续指定所需的任何其他设置，然后选择创建设置以创建新的分配设置资源。

从中为 Image Builder AMI 配置跨账户分发 AWS CLI

本节介绍如何配置分发设置文件以及如何使用中的create-image命令在账户之间构建和分发 Image Builder AMI。 AWS CLI

配置跨账户分配需要特定的 IAM 权限。在运行 create-image 命令之前，您必须完成本部分的 跨账户 AMI 分配的先决条件。

1. 配置分配设置文件

   在使用中的create-image命令创建分发给其他账户的 Image Builder AMI 之前，必须创建一个在`AmiDistributionConfiguration`设置 IDs 中指定目标账户的 `DistributionConfiguration` JSON 结构。 AWS CLI 您必须在源区域中指定至少一个 `AmiDistributionConfiguration`。

   以下名为 `create-distribution-configuration.json` 的示例文件显示了源区域中跨账户映像分配的配置。

```
{
 "name": "cross-account-distribution-example",
 "description": "Cross Account Distribution Configuration Example",
 "distributions": [
  {
   "amiDistributionConfiguration": {
    "targetAccountIds": ["123456789012", "987654321098"],
```

```
    "name": "Name {{ imagebuilder:buildDate }}",
    "description": "ImageCopy Ami Copy Configuration"
   },
   "region": "us-west-2"
  }
 ]
}
```

2. 创建分配设置

   要使用中的create-distribution-configuration命令创建 Image Builder 分发设置资源 AWS CLI，请在命令中提供以下参数：

   - 在 `--name` 参数中输入分配的名称。
   - 附加您在 `--cli-input-json` 参数中创建的分配配置 JSON 文件。

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

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

您也可以使用 *`--distributions`* 参数直接在命令中提供 JSON。

使用 EC2 启动模板配置 AMI 分配

为了帮助确保 Image Builder AMI 在目标账户和区域中获得一致的启动体验，您可以使用 `launchTemplateConfigurations` 在分配设置中指定 Amazon EC2 启动模板。当分配过程中存在 `launchTemplateConfigurations` 时，Image Builder 会创建新版本的启动模板，其中包含模板中的所有原始设置以及来自构建的新 AMI ID。有关使用启动模板启动 EC2 实例的更多信息，请根据您的目标操作系统参阅下列链接之一。

- 通过启动模板启动 Linux 实例
- 通过启动模板启动 Windows 实例

> **Note**
>
> 当您在映像中包含用于启用 Windows 快速启动的启动模板时，启动模板必须包含以下标签，这样 Image Builder 才能代表您启用 Windows 快速启动。
> `CreatedBy: EC2 Image Builder`

通过控制台将 EC2 启动模板添加到 AMI 分配设置

要为您的输出 AMI 提供启动模板，请在控制台中执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择分配设置。这将显示在您的账户下创建的分配设置列表。
3. 在分配设置页面的顶部，选择创建分配设置。这将打开创建分配设置页面。
4. 在映像类型部分，选择 Amazon Machine Image (AMI) 输出类型。这是默认设置。
5. 在常规部分中，输入要创建的分配设置资源的名称（必填）。
6. 在区域设置部分，从列表中选择 EC2 启动模板的名称。如果您的账户中没有启动模板，请选择创建新的启动模板，这将在 EC2 控制面板中打开启动模板。

   选中设置默认版本复选框，将启动模板的默认版本更新为 Image Builder 使用您的输出 AMI 创建的新版本。

   要向所选区域添加其他启动模板，请选择添加启动模板配置。

   要移除启动模板，请选择移除。
7. 继续指定所需的任何其他设置，然后选择创建设置以创建新的分配设置资源。

将 EC2 启动模板添加到 AMI 分发设置中 AWS CLI

本节介绍如何使用启动模板配置分配设置文件，以及如何使用 AWS CLI 中的 create-image 命令来构建和分配 Image Builder AMI 以及使用它的启动模板的新版本。

1. 配置分配设置文件

   在使用启动模板创建 Image Builder AMI 之前 AWS CLI，必须先创建一个用于指定`launchTemplateConfigurations`设置的分发配置 JSON 结构。您必须在源区域中指定至少一个 `launchTemplateConfigurations` 条目。

以下名为 create-distribution-config-launch-template.json 的示例文件显示了在源区域中配置启动模板的几种可能场景。

```
{
    "name": "NewDistributionConfiguration",
    "description": "This is just a test",
    "distributions": [
        {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "test-{{imagebuilder:buildDate}}-
{{imagebuilder:buildVersion}}",
                "description": "description"
            },
            "launchTemplateConfigurations": [
                {
                    "launchTemplateId": "lt-0a1bcde2fgh34567",
                    "accountId": "935302948087",
                    "setDefaultVersion": true
                },
                {
                    "launchTemplateId": "lt-0aaa1bcde2ff3456"
                },
                {
                    "launchTemplateId": "lt-12345678901234567",
                    "accountId": "123456789012"
                }
            ]
        }
    ],
    "clientToken": "clientToken1"
}
```

2. 创建分配设置

要使用中的create-distribution-configuration命令创建 Image Builder 分发设置资源 AWS CLI，请在命令中提供以下参数：

- 在 --name 参数中输入分配的名称。
- 附加您在 --cli-input-json 参数中创建的分配配置 JSON 文件。

```
aws imagebuilder create-distribution-configuration --name my distribution name--
cli-input-json file://create-distribution-config-launch-template.json
```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

您也可以使用 *`--distributions`* 参数直接在命令中提供 JSON。

使用增强的 AMI 分发功能

Image Builder 提供高级分发功能， AMIs 可让您灵活控制跨区域和账户的分布方式。这些功能将分发与构建过程分开，允许您按需分发现有图像，高效地从分发失败中恢复，并通过可自定义的工作流程实施受控的多阶段分发策略。

您可以在 Image Builder 中使用增强的 AMI 分发功能直接执行分发活动，而无需重新运行完整的映像构建。

解耦分布

DistributeImage API 接受三种类型的源图像引用：

- AMI ID — 标准的 AMI 标识符（例如，`ami-0abcdef1234567890`）
- SSM 参数 — 存储 AMI ID 的 SSM 参数（例如，）`ssm:/my/ami/parameter`
- Image Builder 版本 ARN — 图像生成器图像版本 ARN

分发重试

如果图像分发失败，请使用 `RetryImage` API 重试分发。这避免了完整的映像重建，从而缩短了排除故障原因的时间。`RetryImage`在解决分发失败的根本原因后使用。

RetryImage API 接受映像构建版本 ARN（例如）。`arn:aws:imagebuilder:us-west-2:123456789012:image/my-image/1.0.0/1`当您调用 API 时，Image Builder 会使用原始

分发配置和设置自动从故障点恢复分发过程。RetryImageAPI 可以重试在分发阶段、测试阶段或集成阶段失败的分发。它适用于以下状态：待处理、失败、已删除或可用。 AMIs

先决条件

在重试分发之前，请确保满足以下条件：

- 您已经确定并解决了失败的根本原因。查看日志中的分发 CloudWatch 日志，了解错误详情。
- 您拥有重试镜像构建所必需的 IAM 权限。
- 对于跨账户分发失败，请验证目标账户EC2ImageBuilderDistributionCrossAccountRole中的是否已附加Ec2ImageBuilderCrossAccountDistributionAccess策略。

重要：如果不修复根本问题，重试将导致重复失败。

分发工作流程

分发工作流是一种新的工作流类型，它补充了构建和测试工作流程，使您能够通过顺序步骤定义和控制分发流程。通过分发工作流程，您可以创建自定义分发流程，包括 AMI 复制操作、 wait-for-action检查点、图像属性修改和其他与分发相关的步骤。这提供了对分布方式的结构化控制， AMIs 包括分步级可见性、并行分发功能和精细的错误报告。

要了解有关创建和自定义工作流程的更多信息，请参阅管理图像工作流程。

与共享 Image Builder 资源 AWS RAM

EC2 Image Builder 与 AWS Resource Access Manager (AWS RAM) 集成，因此您可以与任何人 AWS 账户 或通过任何人共享以下类型的图像生成器资源 AWS Organizations。

- 组件
- 图片
- 配方

要通过共享资源 AWS RAM，必须创建资源共享。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。消费者可以是个人 AWS 账户、组织单位或中的整个组织 AWS Organizations。以下列表包括您可以与之共享资源的账户和组织类型。

- 具体在其组织 AWS 账户 内部或外部 AWS Organizations。
- AWS Organizations中的组织内部的组织单元 (OU)。
- AWS Organizations中的整个组织。
- AWS Organizations 或在其组织 OUs 之外。

在此模型中 AWS 账户 ，拥有资源的人（所有者）与同一区域内的其他人 AWS 账户 或通过 AWS Organizations （消费者）共享资源。共享资源更新后，使用者会自动获得这些更新。

> **Note**
>
> 共享的组件、映像和映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。

主题

资源所有者

Image Builder 资源只能 AWS 区域 在创建地共享。在共享这些资源时，不会在区域之间复制它们。

要获取您拥有并可以共享的 Image Builder 资源列表，请在控制台中指定所有权筛选条件，或指定何时在 AWS CLI中运行命令。

- 列出 Image Builder 组件
- 列出映像
- 列出和查看映像配方的详细信息
- 列出和查看容器配方详细信息

有关的更多信息 AWS RAM，请参阅《AWS RAM 用户指南》。

共享 Image Builder 资源的先决条件

要共享 Image Builder 资源（例如组件、映像或配方），必须满足以下条件：

- 您 AWS 账户 必须拥有要共享的 Image Builder 资源。您不能共享已与您共享的资源。
- 必须与目标账户、组织或明确共享与加密资源关联的 AWS Key Management Service (AWS KMS) 密钥 OUs。
- 要与您共享 AWS Organizations 并 OUs 使用您的 Image Builder 资源 AWS RAM，您必须启用共享。有关更多信息，请参阅《AWS RAM 用户指南》中的在 AWS Organizations中启用资源共享。
- 如果您在不同区域的账户 AWS KMS 之间分发使用加密的图像，则必须在每个目标区域中创建 KMS 密钥和别名。此外，将在这些地区启动实例的人员需要访问通过密钥策略指定的 KMS 密钥。

Image Builder 通过您的管道构建创建的以下资源不被视为 Image Builder 资源，而是 Image Builder 在您的账户中分配的外部资源，以及您在分配配置中指定的账户、组织或组织单位 (OUs) 的外部资源。AWS 区域

- Amazon 机器映像 (AMIs)
- 存放在 Amazon ECR 中的容器映像

有关适用于 AMI 的分配设置的更多信息，请参阅 创建和更新 AMI 分配配置。有关在 Amazon ECR 中容器映像的分配设置的更多信息，请参阅 创建和更新容器映像的分配设置。

有关与 AWS Organizations 和共享您的 AMI 的更多信息 OUs，请参阅与组织共享 AMI 或 OUs。

资源使用者

使用者可以使用共享资源，但不能以任何方式修改资源。在创建 Image Builder 配方时，他们可以将共享映像指定为基础映像，还可以添加共享组件。他们还可以在创建 Image Builder 映像管道或在 AWS CLI中使用 create-image 命令时指定共享配方。

如果您属于中的某个组织 AWS Organizations，并且启用了组织内的共享，则系统会自动授予您组织中的消费者访问共享资源的权限。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。

为您的 I AWS RAM mage Builder 资源创建资源共享

要共享 Image Builder 组件、图像或配方，必须将其添加到 AWS Resource Access Manager 资源共享中。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。

可以提供以下选项用于共享资源。

选项 1：创建 RAM 资源共享

创建 RAM 资源共享时，只需一步即可共享您所拥有的组件、映像或配方。使用以下方法之一创建资源共享：

- 控制台

  要使用 AWS RAM 控制台创建资源共享，请参阅AWS RAM 用户指南中的共享您拥有的 AWS 资源。
- AWS CLI

  要使用 AWS RAM 命令行界面创建资源共享，请运行中的create-resource-share命令 AWS CLI。

选项 2：应用资源策略并提升到现有的资源共享

共享资源的第二个选项包括两个步骤，两个步骤都在中 AWS CLI 运行命令。第一步使用中的 Image Builder 命令 AWS CLI 将基于资源的策略应用于共享资源。第二步使用中的promote-resource-share-created-from-policy AWS RAM 命令将资源提升为 RAM 资源共享， AWS CLI 以确保与您共享该资源的所有委托人都能看到该资源。

1. 应用资源策略

   要成功应用资源策略，您必须确保与之共享的账户有权访问任何底层资源。

   选择与适用命令的资源类型相匹配的选项卡。

   Image

   您可以将资源策略应用于映像，以允许其他用户在其配方中作为基础映像使用该映像。

   运行中的 put-image-policy Image Builder 命令 AWS CLI，确定要与之共享映像的 AWS 委托人。

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

   Component

   您可以将资源策略应用于构建或测试组件，以启用跨账户共享。此命令为其他账户授予权限，以便在其配方中使用您的组件。要成功应用该资源策略，您必须确保与您共享资源的账户有权访问共享的组件引用的所有资源，例如，在私有存储库上托管的文件。

   运行中的 put-component-policy Image Builder 命令 AWS CLI，确定要与之共享组件的 AWS 委托人。

```
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": [ "123456789012" ] }, "Action":
 [ "imagebuilder:GetComponent", "imagebuilder:ListComponents" ], "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-
component/2019.12.03/1" ] } ] }'
```

Image recipe

您可以将资源策略应用于映像配方，以便启用跨账户共享。此命令为其他账户授予权限，以使用您的配方在其账户中创建映像。要成功应用资源策略，必须确保与之共享的账户有权限访问配方引用的任何资源，如基础映像或选定组件。

运行中的 put-image-recipe-policy Image Builder 命令 AWS CLI，确定要与之共享映像的 AWS 委托人。

```
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": [ "123456789012" ] }, "Action":
 [ "imagebuilder:GetImageRecipe", "imagebuilder:ListImageRecipes" ], "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-image-
recipe/2019.12.03" ] } ] }'
```

Container recipe

您可以将资源策略应用于容器配方，以便启用跨账户共享。此命令为其他账户授予权限，以使用您的配方在其账户中创建映像。要成功应用资源策略，必须确保与之共享的账户有权限访问配方引用的任何资源，如基础映像或选定组件。

运行中的 put-container-recipe-policy Image Builder 命令 AWS CLI，确定要与之共享映像的 AWS 委托人。

```
aws imagebuilder put-container-recipe-policy --container-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-
example-container-recipe/2021.12.03 --policy '{ "Version": "2012-10-17",
        "Statement": [ { "Effect": "Allow", "Principal": { "AWS":
 [ "123456789012" ] }, "Action": [ "imagebuilder:GetContainerRecipe",
 "imagebuilder:ListContainerRecipes" ], "Resource": [ "arn:aws:imagebuilder:us-
west-2:123456789012:container-recipe/my-example-container-
recipe/2021.12.03" ] } ] }'
```

> **Note**
>
> 为了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。

2. 提升为 RAM 资源共享

   要确保与您共享资源的所有委托人都能看到该资源，请运行中的promote-resource-share-created-from-policy AWS RAM AWS CLI命令。

取消共享来自的 Image Builder 资源 AWS RAM

要取消共享您拥有的 Image Builder 资源（例如共享组件、图像或配方），必须将其从 AWS Resource Access Manager 资源共享中移除。您可以使用 AWS RAM 控制台或 AWS CLI以执行该操作。

> **Note**
>
> 除非不再共享资源，否则所有者无法将其删除。所有者不能取消共享这些资源，直到没有使用者依赖它们。

使用控制台取消共享您拥有的共享组件、图像或配方 AWS Resource Access Manager

请参阅《AWS RAM 用户指南》中的更新资源共享。

要取消共享您拥有的共享组件、图像或配方，请使用 AWS CLI

使用 disassociate-resource-share 命令停止共享资源。

标记 Image Builder 输出资源

标记资源有助于筛选和跟踪资源成本或其他类别。您还可以基于标签来控制访问权限。有关基于标签授权的更多信息，请参阅基于 Image Builder 标签的授权

Image Builder 支持以下动态标签：

- `- {{imagebuilder:buildDate}}`

  date/time 在构建时解析为构建。
- `- {{imagebuilder:buildVersion}}`

  解析为构建版本，这是位于 Image Builder Amazon 资源名称 (ARN) 末尾的数字。例如，`"arn:aws:imagebuilder:us-west-2:123456789012:component/myexample-component/2019.12.02/1"` 将构建版本显示为 1。

为了帮助您跟踪已分发的亚马逊系统映像 (AMIs)，Image Builder 会自动将以下标签添加到您的输出中 AMIs。

- `"CreatedBy":"EC2 Image Builder"`
- `"Ec2ImageBuilderArn":"arn:aws:imagebuilder:us-west-2:123456789012:image/simple-recipe-linux/1.0.0/10"`。此标签包含用于创建 AMI 的 Image Builder 映像资源的 ARN。

内容

标记来自的资源 AWS CLI

以下示例展示了如何使用 imagebuilder CLI 命令在 EC2 Image Builder 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
```

从中取消对资源的标记 AWS CLI

以下示例展示了如何使用 imagebuilder CLI 命令从资源中删除标签。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
```

列出来自的特定资源的所有标签 AWS CLI

以下示例展示了如何使用 imagebuilder CLI 命令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
```

删除过期或未使用的 Image Builder 资源

Image Builder 环境就像家一样，需要定期维护，以帮助您找到所需的内容，并在不费吹灰之力的情况下完成任务。请务必定期对为了测试而创建的临时资源进行清理。否则，您可能会忘记这些资源，然后再也不记得它们的用途。届时，可能还不清楚你能否安全地删除它们。

删除资源不会删除在映像构建过程中创建的任何 Amazon EC2 AMI 或 Amazon ECR 容器映像。您必须使用相应的 Amazon EC2 或 Amazon ECR 控制台操作或 API 或 AWS CLI 命令单独清理这些内容。

> **Tip**
>
> 为防止在删除资源时出现依赖项错误，请确保按以下顺序删除资源：
>
> 1. 映像管道
> 2. 映像配方
> 3. 所有剩余的资源

从中删除资源 AWS 管理控制台

若要删除映像管道及其资源，请按照下列步骤操作：

删除管道

1. 要查看在您的账户下创建的构建管道列表，请从导航窗格中选择映像管道。
2. 导航到 镜像管道 页面，然后选中要删除的管道名称旁边的复选框。
3. 在映像管道面板顶部的操作菜单中，选择删除。
4. 若要确认删除，请在框中输入 `Delete`，然后选择删除。

删除配方

1. 要查看在您的账户下创建的配方列表，请从导航窗格中选择映像配方。
2. 选择配方名称旁边的复选框以选择想要删除的配方。
3. 在映像配方面板顶部的操作菜单中，选择删除配方。
4. 若要确认删除，请在框中输入 `Delete`，然后选择删除。

删除基础设施配置

1. 要查看在您的账户下创建的基础设施配置列表，请从导航窗格中选择基础设施配置。
2. 选择配置名称旁边的复选框以选择想要删除的基础架构配置。
3. 在基础设施配置面板的顶部，选择删除。
4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除分配设置

1. 要查看在您的账户下创建的分配设置列表，请从导航窗格中选择分配设置。
2. 选择配置名称旁边的复选框以选择您为本教程创建的分配设置。
3. 在分配设置面板的顶部，选择删除。
4. 若要确认删除，请在框中输入 Delete，然后选择删除。

删除映像

1. 要查看在您的账户下创建的映像列表，请从导航窗格中选择映像。
2. 对于要移除的映像，选择映像版本。此时将打开映像构建版本页面。
3. 选中要删除的任何图像的版本旁边的复选框。您一次可以选择多个映像版本。
4. 在映像构建版本面板的顶部，选择删除版本。
5. 若要确认删除，请在框中输入 Delete，然后选择删除。

从中删除图像管道 AWS CLI

以下示例说明如何使用 AWS CLI删除 Image Builder 资源。如前所述，必须按以下顺序删除资源以避免依赖项错误：

1. 映像管道
2. 映像配方
3. 所有剩余的资源

从中删除图像管道 AWS CLI

以下示例说明了如何指定 ARN 以删除镜像管道。

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

从中删除图片配方 AWS CLI

以下示例说明了如何指定 ARN 以删除镜像配方。

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

删除基础设施配置

以下示例说明了如何指定 ARN 以删除基础设施配置资源。

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

删除分配设置

以下示例说明了如何指定 ARN 以删除分配设置资源。

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

删除映像

以下示例说明了如何指定 ARN 以删除镜像生成版本。

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

删除组件

以下示例演示如何使用 imagebuilder CLI 命令，通过指定其 ARN 来删除组件构建版本。

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

Important

在删除组件构建版本之前，请确保没有任何配方以任何方式引用该版本。否则，可能会导致管道故障。

管理 Image Builder 映像的构建和测试工作流

图像工作流程定义了 Im EC2 age Builder 在图像创建过程的构建、测试和分发阶段执行的步骤顺序。这是整个 Image Builder 工作流框架的一部分。

映像工作流优势

- 借助映像工作流，您可以更灵活、更清晰和更好地控制映像创建过程。
- 您可以在定义工作流文档时添加自定义工作流步骤，也可以选择使用 Image Builder 默认工作流。
- 您可以排除默认映像工作流中包含的工作流步骤。
- 您可以创建完全跳过构建过程的仅测试工作流。您也可以这样做来创建仅限编译或仅限分发的工作流程。

> **Note**
>
> 您无法修改现有工作流，但可以对其进行克隆或创建新版本。

工作流程主题

工作流框架：阶段

要自定义映像工作流，了解构成映像创建工作流框架的工作流阶段至关重要。

图像创建工作流程框架包括以下不同的阶段。

1. 构建阶段（预快照）— 在构建阶段，您可以对运行您的基础映像的 Amazon EC2 构建实例进行更改，以创建新映像的基准。例如，您的配方可以包含安装应用程序或修改操作系统防火墙设置的组件。

成功完成此阶段后，Image Builder 会创建快照或容器映像，以供测试期和后续时期使用。

2. 测试阶段（快照后）— 在测试阶段，创建镜像 AMIs 和容器镜像之间存在一些差异。对于 AMI 工作流程，Image Builder 从作为构建阶段最后一步创建的快照启动 EC2 实例。在新实例上运行测试以验证设置并确保该实例按预期运行。对于容器工作流，测试在用于构建的同一实例上运行。
3. 分发阶段（后期构建）— 在分发阶段，一旦映像构建完成并成功通过所有测试，输出图像就会经历不同的分发阶段。这些阶段包括 AMI 复制操作、图像属性修改、图像共享和其他相关的分发步骤。

服务访问

要运行映像工作流，Image Builder 需要获得执行工作流操作的权限。您可以指定 AWSServiceRoleForImageBuilder 服务相关角色，也可以为服务访问指定自己的自定义角色，如下所示。

- 控制台 – 在管道向导步骤 3 定义映像创建过程中，从服务访问面板的 IAM 角色列表中选择服务相关角色或您自己的自定义角色。
- I@@ mage Builder API — 在CreateImage操作请求中，将服务相关角色或您自己的自定义角色指定为`executionRole`参数值。

要详细了解如何创建服务角色，请参阅《AWS Identity and Access Management 用户指南》中的创建角色以向 AWS 服务委派权限。

对图像使用托管工作流程

托管工作流程由创建和维护 AWS。当您在图像管道中使用托管工作流程或创建一次性图像时，您可以选择要使用的托管工作流程的 Amazon 资源名称 (ARN)。Amazon 提供已应用补丁和其他更新的最新版本。要获取托管工作流程列表列出映像工作流，请参阅并在 Owner = Amazon（控制台）上进行筛选。

列出映像工作流

在 Image Builder 控制台的映像工作流列表页面上，您可以获取您拥有或有权访问的映像工作流资源列表，以及有关这些资源的一些关键详细信息。您还可以在 Image Builder API 中使用命令或操作 SDKs，或者 AWS CLI 在您的账户中列出图像工作流程。

您可以使用以下方法之一列出您拥有或有权访问的映像工作流资源。有关 API 操作，请参阅 I EC2 mage Builder API 参考ListWorkflows中的。有关关联的 SDK 请求，请参阅同一页面上的另请参阅链接。

您可以根据以下详细信息进行筛选，以简化结果列表。例如，如果您在控制台中对 Own er = Amazon 进行筛选，则列表将仅显示 AWS 托管工作流程。

- 工作流
- 版本
- Type
- 所有者

Console

工作流详细信息

Image Builder 控制台中映像工作流列表页面上的详细信息包括以下内容：

- 工作流 – 映像工作流资源最新版本的名称。在 Image Builder 控制台中，工作流列链接到工作流详细信息页面。
- 版本 – 映像工作流资源的最新版本。
- 类型-工作流程类型：BUILDTEST、或DISTRIBUTION。
- 所有者 – 工作流资源的所有者。
- 创建日期 – Image Builder 创建最新版本映像工作流资源的日期和时间。
- ARN – 当前版本映像工作流资源的 Amazon 资源名称（ARN）。

列出映像工作流

要在 Image Builder 控制台中列出映像工作流资源，请执行以下步骤：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择映像工作流。

筛选结果

在映像工作流列表页面上，您可以搜索特定的映像工作流来筛选结果。以下筛选条件可用于映像工作流：

`Workflow`

您可以输入完整或部分工作流名称以简化结果。默认为在列表中显示所有工作流。

`Version`

您可以输入完整或部分版本号以简化结果。默认为在列表中显示所有版本。

`Type`

您可以按工作流类型进行筛选，也可以查看所有类型。默认为在列表中显示所有工作流类型。

- BUILD
- 测试
- 分布

`Owner`

当您从搜索栏中选择所有者筛选条件时，Image Builder 会显示您账户中映像工作流的所有者列表。您可以从列表中选择所有者以简化结果。默认为显示列表中的全部所有者。

- AWS 账户 – 拥有工作流资源的账户。
- Amazon – Amazon 拥有和管理的工作流资源。

AWS CLI

在中运行list-workflows命令时 AWS CLI，您可以获得自己拥有或有权访问的图像工作流程列表。

以下命令示例演示了如何使用不带筛选条件的 list-workflows 命令列出您拥有或有权访问的所有 Image Builder 映像工作流资源。

示例：列出所有映像工作流

```
aws imagebuilder list-workflows
```

输出：

```
{
    "workflowVersionList": [
        {
            "name": "example-test-workflow",
```

```
            "dateCreated": "2023-11-21T22:53:14.347Z",
            "version": "1.0.0",
            "owner": "111122223333",
            "type": "TEST",
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
test/example-test-workflow/1.0.0"
        },
        {
            "name": "example-build-workflow",
            "dateCreated": "2023-11-20T12:26:10.425Z",
            "version": "1.0.0",
            "owner": "111122223333",
            "type": "BUILD",
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
build/example-build-workflow/1.0.0"
        },
        {
            "name": "example-distribution-workflow",
            "dateCreated": "2025-11-19T10:25:10.425Z",
            "version": "1.0.0",
            "owner": "111122223333",
            "type": "DISTRIBUTION",
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
distribution/example-distribution-workflow/1.0.0"
        }
    ]
}
```

运行 list-workflows 命令时，可以应用筛选器来简化结果，如以下示例所示。有关如何筛选结果的更多信息，请参阅《AWS CLI Command Reference》中的 list-workflows 命令。

示例：构建工作流的筛选条件

```
aws imagebuilder list-workflows --filters name="type",values="BUILD"
```

输出：

```
{
    "workflowVersionList": [
        {
            "name": "example-build-workflow",
            "dateCreated": "2023-11-20T12:26:10.425Z",
            "version": "1.0.0",
```

```
            "owner": "111122223333",
            "type": "BUILD",
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
build/example-build-workflow/1.0.0"
        }
    ]
}
```

创建映像工作流

创建映像工作流时，您可以更好地控制映像创建过程。您可以指定 Image Builder 构建和测试映像时运行的工作流。您还可以指定客户托管密钥以加密工作流资源。要了解有关工作流资源加密的更多信息，请参阅Image Builder 中的加密和密钥管理。

对于映像创建，您可以指定一个构建阶段工作流以及一个或多个测试阶段工作流。根据需求，您甚至可以完全跳过构建或测试阶段。您可以在工作流使用的 YAML 定义文档中配置工作流执行的操作。有关 YAML 文档语法的更多信息，请参阅创建 YAML 工作流文档。

有关创建新构建或测试工作流的步骤，请选择与您将要使用的环境相匹配的选项卡。

AWS 管理控制台

您可以按照以下步骤在 Image Builder 控制台中创建新的工作流。

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 从导航窗格中，选择映像工作流。此操作将显示您账户拥有或有权访问的映像工作流列表。

   > **Note**
   >
   > 您将始终在列表中看到 Image Builder 用于其默认工作流、由 Amazon 管理的工作流资源。要查看这些工作流的详细信息，可以选择工作流链接。

3. 要创建新的工作流，请选择创建映像工作流。此操作将显示创建映像工作流页面。
4. 为新工作流配置详细信息。要创建构建工作流，请选择表单顶部附近的构建选项。要创建测试工作流，请选择表单顶部附近的测试选项。Image Builder 会根据此选项填充模板列表。构建和测试工作流的所有其他步骤都相同。

   一般性问题

   “常规”部分包括适用于工作流资源的设置，例如名称和描述。“常规”设置包含以下内容：

- 映像工作流名称（必需）– 映像工作流的名称。该名称在您的账户中必须是唯一的。名称长度不超过 128 个字符。有效字符包括字母、数字、空格、- 和 _。
- 版本（必需）– 要创建的工作流资源的语义版本（major.minor.patch）。
- 描述（可选）– 可以选择性为工作流添加描述。
- KMS 密钥（可选）– 您可以使用客户托管密钥加密工作流资源。有关更多信息，请参阅 使用客户托管密钥加密映像工作流。

定义文档

YAML 工作流文档包含工作流的所有配置。

开始使用

- 要开始使用 Image Builder 默认模板作为工作流的基准，请选择从模板开始选项。已默认选定此选项。从模板列表中选择要使用的模板后，会将您选择的模板中的默认配置复制到新工作流文档的内容中，您可以在其中进行更改。
- 要从头开始定义工作流文档，请选择从头开始选项。此操作会在内容中填充文档格式中一些重要部分的简短概述，以帮助您入门。

内容面板底部有一个状态栏，显示 YAML 文档的警告或错误。有关如何创建 YAML 工作流文档的更多信息，请参阅创建 YAML 工作流文档。

5. 完成工作流后，或者如果您想保存进度稍后返回，请选择创建工作流。

AWS CLI

在中运行create-workflow命令之前 AWS CLI，必须创建包含工作流程所有配置的 YAML 文档。有关更多信息，请参阅 创建 YAML 工作流文档。

以下示例演示了如何使用 create-workflow AWS CLI 命令创建构建工作流。`--data` 参数是指 YAML 文档，其中包含您创建的工作流的构建配置。

示例：创建工作流

```
aws imagebuilder create-workflow --name example-build-workflow --semantic-
version 1.0.0 --type BUILD --data file://example-build-workflow.yml
```

输出：

```
{
 "workflowBuildVersionArn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
build/example-build-workflow/1.0.0/1",
 "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

以下示例演示了如何使用 create-workflow AWS CLI 命令创建测试工作流。--data 参数是指 YAML 文档，其中包含您创建的工作流的构建配置。

示例：创建测试工作流

```
aws imagebuilder create-workflow --name example-test-workflow --semantic-
version 1.0.0 --type TEST --data file://example-test-workflow.yml
```

输出：

```
{
 "workflowBuildVersionArn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
test/example-test-workflow/1.0.0/1",
 "clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

创建 YAML 工作流文档

YAML 格式定义文档为映像构建过程的构建和测试阶段配置输入、输出和工作流步骤。您可以从包含标准化步骤的模板开始，也可以从头开始定义自己的工作流。无论您使用模板还是从头开始，都可以自定义工作流以满足您的需求。

YAML 工作流文档的结构

Image Builder 用于执行映像构建和测试操作的 YAML 工作流文档的结构如下。

- 工作流文档标识
- 工作流文档输入参数
- 工作流文档步骤
- 工作流文档输出

工作流文档标识

对工作流进行唯一标识。此部分可以包含以下属性。

字段	描述	Type	必需
name	工作流文档名称。	字符串	否
描述	文档描述。	字符串	否
schemaVersion	文档架构版本，当前为 1.0。	字符串	是

示例

```
---
name: sample-test-image
description: Workflow for a sample image, with extra configuration options exposed
 through workflow parameters.
schemaVersion: 1.0
```

工作流文档输入参数

工作流文档的这一部分定义了调用方可以指定的输入参数。如果您没有任何参数，则可以省略此部分。如果您确实指定了参数，则每个参数可以包含以下属性。

字段	描述	Type	必需	约束
name	参数的名称。	字符串	是	
描述	参数描述。	字符串	否	
默认	如果未提供值，则为该参数的默	匹配参数数据类型。	否	

字段	描述	Type	必需	约束
	认值。如果参数定义中未包含默认值，则运行时需要该参数值。			
类型	参数的数据类型。如果参数定义中未包含数据类型，则参数类型默认为运行时所需的字符串值。	字符串	是	参数的数据类型必须为以下类型之一： • `string` • `integer` • `boolean` • `stringList`

示例

在工作流文档中指定参数。

```
parameters:
  - name: waitForActionAtEnd
    type: boolean
    default: true
    description: "Wait for an external action at the end of the workflow"
```

在工作流文档中使用参数值。

```
$.parameters.waitForActionAtEnd
```

工作流文档步骤

为工作流指定最多 15 个步骤操作。步骤按照工作流文档中定义的顺序运行。如果失败，将按相反的顺序运行回滚，从失败的步骤开始，然后反向执行之前的步骤。

每个步骤都可以引用任何先前步骤操作的输出。这称为链接或引用。要引用前一步操作的输出，可以使用 JSONPath 选择器。例如：

```
$.stepOutputs.step-name.output-name
```

有关更多信息，请参阅 在工作流文档中使用动态变量。

> **Note**
>
> 尽管步骤本身并没有输出属性，但步骤操作的任何输出都包含在该步骤的 `stepOutput` 中。

每个步骤都可以包含以下属性。

字段	描述	Type	必需	默认 值	约束
action	此步骤执行的工作流操作。	字符串	是		必须是 Image Builder 工作流文档支持的步骤操作。
`if`，后跟一组修改 `if` 运算符的条件语句。	条件语句将控制决策点流程添加到工作流步骤的主体中。	字典	否		Image Builder 支持以下条件语句作为 `if` 运算符的修饰符： • 分支条件和修饰符：`if`、`and`、`or`、`not`。分支条件在某一行上自行指定。 • 比较运算符：`booleanEq`

字段	描述	Type	必需	默认 值	约束
					uals 、numberEq als 、numberGre aterThan 、numb aterThanE quals 、numberL sThan 、numberL sThanEqua ls 、stringEqu als 。
描述	步骤描述。	字符串	否		不允许使用空字符串。如果包含，长度必须为 1-1024 个字符。
输入	包含步骤操作运行所需的参数。您可以将键值指定为静态值，也可以使用可解析为正确数据类型的 JSONPath 变量来指定。	字典	是		

字段	描述	Type	必需	默认 值	约束
name	步骤的名称。在工作流文档内，此名称必须具有唯一性。	字符串	是		长度必须介于 3-128 个字符之间。 可以包含字母数字字符和 _。不能包含空格。

字段	描述	Type	必需	默认 值	约束
onFailure	配置步骤失败时要执行的操作，如下所示。 行为 • Abort – 步骤失败，工作流失败，并且在失败的步骤之后不运行任何其余步骤。如果启用了回滚，则从失败的步骤开始回滚，一直持续到允许回滚的所有步骤都回滚为止。 • Continue – 步骤失败，但在失败的步骤之后继续运行其余步骤。在这种情况下，不会进行回滚。	字符串	否	Abort	Abort | Continue

字段	描述	Type	必需	默认 值	约束
rollbackEnabled	配置发生故障时是否回滚该步骤。您可以使用静态布尔值或解析为布尔值的动态 JSONPath 变量。	布尔值	否	`true`	`true` | `false` | 或者解析为真或假的 JSONPath 变量。
timeoutSeconds	在失败和重试（如果重试适用）之前步骤运行的最长时间（以秒为单位）。	整数	否	取决于为步骤操作定义的默认值（如果适用）。	不能超过步骤操作的最大超时时间
等待秒	步骤执行暂停的时间，以秒为单位。	整数	否	0	不能超过步骤操作的 timeoutSeconds

示例

```
steps:
  - name: LaunchTestInstance
    action: LaunchInstance
    onFailure: Abort
    inputs:
      waitFor: "ssmAgent"

  - name: ApplyTestComponents
    action: ExecuteComponents
    onFailure: Abort
    inputs:
```

```
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"

  - name: TerminateTestInstance
    action: TerminateInstance
    onFailure: Continue
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"

  - name: WaitForActionAtEnd
    action: WaitForAction
    if:
      booleanEquals: true
      value: "$.parameters.waitForActionAtEnd"
```

工作流文档输出

定义工作流的输出。每个输出都是一个键值对，可指定输出的名称和值。您可以使用输出在运行时导出后续工作流可以使用的数据。此部分是可选的。

您定义的每个输出都包含以下属性。

字段	描述	Type	必需
name	输出的名称。此名称在管道包含的工作流中必须具有唯一性。	字符串	是
值	输出的值。字符串的值可以是动态变量，例如步骤操作的输出文件。有关更多信息，请参阅 在工作流文档中使用动态变量。	字符串	是

示例

使用 `createProdImage` 步骤的步骤输出为工作流文档创建输出映像 ID。

```
outputs:
  - name: 'outputImageId'
    value: '$.stepOutputs.createProdImage.imageId'
```

请参阅下一个工作流中的工作流输出。

```
$.workflowOutputs.outputImageId
```

工作流程文档支持的步骤操作

本节详细介绍了 Image Builder 支持的步骤操作。

本节中使用的术语

AMI

亚马逊机器映像

进行筛选

Amazon 资源名称

支持的操作

- ApplyImageConfigurations
- BootstrapInstanceForContainer
- CollectImageMetadata
- CollectImageScanFindings
- CreateImage
- DistributeImage
- ExecuteComponents
- ExecuteStateMachine
- LaunchInstance
- ModifyImageAttributes
- RegisterImage
- RunCommand

- RunSysPrep
- SanitizeInstance
- TerminateInstance
- WaitForAction
- WaitForSSMAgent

ApplyImageConfigurations

此步骤操作将各种配置和集成应用于分布式 AMIs，例如许可证配置、启动模板配置、S3 导出配置、EC2 快速启动配置和 Systems Manager 参数配置。配置仅适用于源账户中的分布式映像，但可以跨账户应用的 SSM 参数配置除外。

默认超时：360 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
region	图像区域。	字符串	是		
licenseConfigurationArns	镜像的许可配置 ARN。	数组	否		
launchTemplateConfigurations		数组	否		
launchTemplateConfiguration s:launchTemplateId	要应用于图像的启动模板 ID。	字符串	如果`launchTemplateConfigurations` 已指定，则是		

输入名称	说明	Type	必需	默认	约束
launchTemplateConfigurations:账户 ID	IDs 要应用于图片的启动模板账户。	字符串	否		
launchTemplateConfigurations:setDefaultVersion	图像的启动模板默认版本设置。	布尔值	否		
s3 ExportConfiguration		数组	否		
s3ExportConfiguration:角色名	映像的 S3 导出配置角色名称。	字符串	如果`s3ExportConfiguration` 已指定，则是		
s3ExportConfiguration：diskImageFormat	映像的 S3 导出配置磁盘映像格式。	字符串	如果`s3ExportConfiguration` 已指定，则是		允许的值-VMDK|RAW|VHD
s3: s3BucketExportConfiguration	映像的 S3 导出配置存储桶名称。	字符串	如果`s3ExportConfiguration` 已指定，则是		
s3: s3PrefixExportConfiguration	映像的 S3 导出配置存储桶前缀。	字符串	否		

输入名称	说明	Type	必需	默认	约束
fastLaunchConfigurations	映像的 EC2 快速启动配置。	数组	否		
fastLaunchConfigurations:已启用	EC2 图像 enabled/disabled 的快速启动。	布尔值	如果`fastLaunchConfigurations` 已指定，则是		
fastLaunchConfigurations:快照配置	EC2 图像 enabled/disabled 的快速启动。	Map	否		
fastLaunchConfigurations:快照配置：targetResourceCount	EC2 快速启动图像的目标资源数量。	整数	否		
fastLaunchConfigurations:maxParallelLaunches	EC2 Fast Launch 图像的最大并行启动次数。	整数	否		
fastLaunchConfigurations:启动模板			否		

输入名称	说明	Type	必需	默认	约束
fastLaunchConfigurations:启动模板:launchTemplateId	EC2 图像的快速启动启动模板 ID。	字符串	否		
fastLaunchConfigurations:启动模板:launchTemplateName	EC2 快速启动镜像的启动模板名称。	字符串	否		
fastLaunchConfigurations:启动模板:launchTemplateVersion	EC2 快速启动启动镜像的模板版本。	字符串	否		
ssmParameterConfigurations	镜像的 SSM 参数配置。	Map	否		
ssmParameterConfigurations:amiAccountId	图像的 SSM 参数 AMI 账户 ID。	字符串	否		
ssmParameterConfigurations:parameterName	图像的 SSM 参数名称。	字符串	如果`ssmParameterConfigurations` 已指定，则是		

输入名称	说明	Type	必需	默认	约束
ssmParameterConfigurations: dataType	图像的 SSM 参数数据类型。	字符串	否		允许的值-文本 | aws: ec2: image)

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
配置镜像	已配置映像的列表。	数组
配置镜像：账号 ID	分布式镜像的目标账户 ID。	字符串
配置图像:名称	AMI 的名称。	字符串
配置图片:amiid	分布式映像的 AMI ID。	字符串
配置图片：开始日期	开始分发的 UTC 时间。	字符串
配置图片：日期已停止	分发完成时的 UTC 时间。	字符串
配置图像:步骤	分发停止的步骤。	已完成| | AssociateLicensesRunning | | UpdateLaunchTemplateRunning | PutSsmParametersRunning | UpdateFastLaunchConfiguration ExportAmiQueued ExportAmiRunning
配置图片:区域	分布式映像的 AWS	字符串
配置图像:状态	分发状态。	已完成 | 失败 | 已取消 | TimedOut
配置图像：错误消息	错误消息（如果有）。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: ApplyImageConfigurations
  action: ApplyImageConfigurations
  onFailure: Abort
  inputs:
    distributedImages.$: $.stepOutputs.DistributeImageStep.distributedImages
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.ApplyImageConfigurationsStep.configuredImages
```

BootstrapInstanceForContainer

此步骤操作运行服务脚本来引导实例，以最低要求运行容器工作流。Image Builder 使用 Systems Manager API 中的 sendCommand 来运行此脚本。有关更多信息，请参阅 AWS Systems Manager Run Command。

> **Note**
>
> 引导脚本会安装 AWS CLI 和 Docker 包，它们是 Image Builder 成功构建 Docker 容器的先决条件。如果未包含此步骤操作，映像构建可能会失败。

默认超时：60 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要引导的实例 ID。	字符串	是		这必须是启动此工作流实例的工作流步骤

输入名称	说明	Type	必需	默认	约束
					的输出实例 ID。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
runCommandId	在实例上运行引导脚本的 Systems Manager sendCommand 的 ID。	字符串
status	从 Systems Manager sendCommand 返回的状态。	字符串
output	从 Systems Manager sendCommand 返回的输出。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: ContainerBootstrapStep
  action: BootstrapInstanceForContainer
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.ContainerBootstrapStep.status
```

CollectImageMetadata

此步骤操作仅对构建工作流有效。

EC2 Image Builder 在其启动的 EC2 实例上运行 AWS Systems Manager（Systems Manager）代理，以构建和测试您的映像。Image Builder 使用 Systems Manager 清单，以收集在构建阶段使用的

实例的其他信息。该信息包括操作系统 (OS) 名称和版本，以及操作系统报告的软件包及其相应版本的列表。

> **Note**
>
> 此步骤操作仅适用于创建的图像 AMIs。

默认超时：30 分钟

最大超时：720 分钟

回滚：Image Builder 会回滚在此步骤中创建的所有 Systems Manager 资源。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	应用元数据设置的构建实例。	字符串	是		这必须是启动此工作流构建实例的工作流步骤的输出实例 ID。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
osVersion	从构建实例收集的操作系统名称和版本。	字符串
associationId	用于清单收集的 Systems Manager 关联 ID。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: CollectMetadataStep
  action: CollectImageMetadata
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作的输出。

```
$.stepOutputs.CollectMetadataStep.osVersion
```

CollectImageScanFindings

如果您的账户启用了 Amazon Inspector，并且您的管道启用了映像扫描，则此步骤操作会收集 Amazon Inspector 为您的测试实例报告的映像扫描调查发现。此步骤操作不适用于构建工作流。

默认超时：120 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	在其上运行扫描的实例的 ID。	字符串	是		这必须是启动此工作流实例的工作流步骤的输出实例 ID。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
runCommandId	运行脚本以收集调查发现的 Systems Manager sendCommand 的 ID。	字符串

输出名称	说明	Type
status	从 Systems Manager sendCommand 返回的状态。	字符串
output	从 Systems Manager sendCommand 返回的输出。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: CollectFindingsStep
  action: CollectImageScanFindings
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.CollectFindingsStep.status
```

CreateImage

此步骤操作使用 Amazon EC2 `CreateImage` API 从正在运行的实例创建映像。在创建过程中，步骤操作会根据需要进行等待，以验证资源是否已达到正确的状态，然后再继续后续操作。

默认超时：720 分钟

最大超时时间：3 天

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要从中创建新映像的实例。	字符串	是		此步骤开始时，提供实例 ID 的实

输入名称	说明	Type	必需	默认	约束
					例必须处于 `running` 状态。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
imageId	所创建映像的 AMI ID。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: CreateImageFromInstance
  action: CreateImage
  onFailure: Abort
  inputs:
      instanceId.$: "i-1234567890abcdef0"
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.CreateImageFromInstance.imageId
```

DistributeImage

此步骤操作将 AMI 分发到指定的区域和账户。它根据工作流程中提供的 CreateImage CreateImagePipeline APIs或和/或自定义分发设置请求中提供的分配配置在目标区域和账户中创建 AMI 的副本，以覆盖分配配置中的设置。

默认超时：360 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
region	要分发图像的区域列表。	字符串	是		最小长度为 1。最大长度为 1024。
name	分发配置的名称。	字符串	否		
描述	分布配置的分布。	字符串	否		
targetAccountIds	IDs 要将图像分发到的账户。	数组	否		
amiTags	分发配置的标签。	Map	否		
kmsKeyId	应用于分布式映像的 KMS 密钥。	字符串	否		

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
分布式图片	分布式映像列表	数组
分布式图片:区域	分布式图像的 AWS 区域。	字符串
分布式图片:名称	AMI 的名称。	字符串
分布式图片:amiid	分布式映像的 AMI ID。	字符串
分布式图片：账户 ID	分布式镜像的目标账户 ID。	字符串
分布式图片：开始日期	开始分发的 UTC 时间。	字符串

输出名称	说明	Type
分布式图片：日期已停止	分发完成时的 UTC 时间。	字符串
分布式图片:状态	分发状态。	已完成 | 失败 | 已取消 | TimedOut
分布式图像:步骤	分发停止的步骤。	已完成| CopyAmiRunning
分布式镜像：错误消息	错误消息（如果有）。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: DistributeImage
  action: DistributeImage
  onFailure: Abort
  inputs:
    distributions:
      - region.$: "$.parameters.SourceRegion"
        description: "AMI distribution to source region"
        amiTags:
          DistributionTest: "SourceRegion"
          WorkflowStep: "DistributeToSourceRegion"
          BuildDate: "{{imagebuilder:buildDate:yyyyMMHHss}}"
          BuildVersion: "{{imagebuilder:buildVersion}}"
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.DistributeImageStep.distributedImages
```

ExecuteComponents

此步骤操作会运行配方中为当前正在构建的映像指定的组件。构建工作流在构建实例上运行构建组件。测试工作流仅在测试实例上运行测试组件。

Image Builder 使用 Systems Manager API 中的 sendCommand 来运行组件。有关更多信息，请参阅 AWS Systems Manager Run Command。

默认超时：720 分钟

最大超时时间：1 天

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	组件应在其上运行的实例的 ID。	字符串	是		这必须是启动此工作流实例的工作流步骤的输出实例 ID。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
runCommandId	在实例上运行组件的 Systems Manager sendCommand 的 ID。	字符串
status	从 Systems Manager sendCommand 返回的状态。	字符串
output	从 Systems Manager sendCommand 返回的输出。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: ExecComponentsStep
  action: ExecuteComponents
  onFailure: Abort
```

```
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作的输出。

```
$.stepOutputs.ExecComponentsStep.status
```

ExecuteStateMachine

此步骤操作从 Image Builder 工作流程中开始执行 AWS Step Functions 状态机。Image Builder 使用 Step Functions `StartExecution` API 启动状态机并等待状态机完成。这对于将复杂的工作流程、合规性验证或认证流程集成到您的图像构建管道中非常有用。

有关更多信息，请参阅《AWS Step Functions 开发者指南》中的 Step Functions 中了解状态机。

默认超时：6 小时

最大超时：24 小时

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
stateMachineArn	要执行的 Step Functions 状态机的 ARN。	字符串	是		必须是有效的状态机 ARN。
input	要提供给状态机的 JSON 输入数据。	字符串	否	{}	必须是有效的 JSON 字符串，最大长度：16 KiB。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
执行 Arn	状态机执行的 ARN。	字符串
output	状态机执行的输出。	字符串

需要 IAM 权限

您的自定义执行角色必须具有以下权限才能使用此步骤操作：

允许操作

- `states:StartExecution`
- `states:DescribeExecution`

指定资源

- `arn:aws:states:us-west-2:111122223333:stateMachine:state-machine-name`
- `arn:aws:states:us-west-2:111122223333:execution:state-machine-name:*`

示例

在工作流文档中指定步骤操作。

```
- name: ValidateImageCompliance
  action: ExecuteStateMachine
  timeoutSeconds: 3600
  onFailure: Abort
  inputs:
    stateMachineArn: arn:aws:states:us-
west-2:111122223333:stateMachine:ImageComplianceValidation
    input: |
      {
        "imageId": "{{ $.stepOutputs.CreateImageFromInstance.imageId }}",
        "region": "us-west-2",
        "complianceLevel": "high",
        "requiredScans": ["cve", "benchmark", "configuration"]
      }
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.ValidateImageCompliance.executionArn
```

LaunchInstance

此步骤操作将在您的中启动一个实例， AWS 账户 并等待 Systems Manager 代理在该实例上运行，然后再继续下一步操作。启动操作使用与您的映像关联的配方和基础设施配置资源中的设置。例如，要启动的实例类型来自基础设施配置。输出是其启动的实例的实例 ID。

`waitFor` 输入会配置满足步骤完成要求的条件。

默认超时：75 分钟

最大超时：720 分钟

回滚：对于构建实例，回滚会执行您在基础设施配置资源中配置的操作。默认情况下，如果映像创建失败，则会终止构建实例。但是，基础设施配置中有一个设置，用于保留构建实例以进行故障排除。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
imageIdOverride	用于启动实例的镜像	字符串	否	构建阶段：图像配方基础图片 测试阶段：从构建阶段输出 AMI	必须是有效的 AMI ID
instanceTypesOverride	Image Builder 会尝试列表中的每种实例类型，直到找到成功启动的实例类型	字符串列表	否	在您的基础设施配置中指定的实例类型	必须是有效的实例类型

输入名称	说明	Type	必需	默认	约束
waitFor	在完成工作流程步骤并进入下一步之前要等待的条件	字符串	是		Image Builder 支持 `ssmAgent`。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
instanceId	所启动实例的实例 ID。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: LaunchStep
  action: LaunchInstance
  onFailure: Abort
  inputs:
    waitFor: ssmAgent
```

在工作流文档中使用步骤操作的输出。

```
$.stepOutputs.LaunchStep.instanceId
```

ModifyImageAttributes

此步骤操作修改分布式的属性 AMIs，例如启动权限和其他 AMI 属性。它对 AMIs 已分配给目标地区和账户的内容进行操作。

默认超时：120 分钟

最大超时时间：180 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
region	图像的区域。	字符串	是		
启动许可			否		
启动权限：用户 ID	在图像 IDs 的启动权限中要修改的用户。	字符串	否		
启动权限：用户组	在图像的启动权限中要修改的用户组。	字符串	否		
启动权限：组织 ARN	在镜像的启动权限中 ARNs 要修改的 AWS 组织。	字符串	否		
启动权限：organizationalUnitArns	在镜像的启动权限中 ARNs 要修改的 AWS 组织单位。	字符串	否		

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
ModifiedImages	修改过的图像列表	数组
修改过的图片:账号 ID	分布式镜像的目标账户 ID。	字符串
修改后的图片:名称	AMI 的名称。	字符串
修改后的图片:amiid	分布式映像的 AMI ID。	字符串

输出名称	说明	Type
修改后的图片:开始日期	开始分发的 UTC 时间。	字符串
修改后的图片:日期已停止	分发完成时的 UTC 时间。	字符串
修改后的图像:步骤	分发停止的步骤。	已完成| ModifyAmiRunning
修改后的图片:区域	图像的 AWS 区域。	字符串
修改后的图片:状态	分发状态。	已完成 | 失败 | 已取消 | TimedOut
修改后的图像：错误消息	错误消息（如果有）。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: ModifyImageAttributes
  action: ModifyImageAttributes
  onFailure: Abort
  inputs:
    distributedImages.$: $.stepOutputs.DistributeImageStep.distributedImages
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.ModifyImageAttributesStep.modifiedImages
```

RegisterImage

此步骤操作使用亚马逊 API 注册新的亚马逊系统映像 (AM EC2 RegisterImage I)。它允许您根据现有快照或一组快照创建 AMI，并指定各种图像属性。

默认超时：540 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
架构	AMI 的架构。	字符串	否		有效值：i386、x86_64、arm64、x86_64_mac、arm64_mac
blockDeviceMapping	AMI 的块储存设备映射条目。	数组	否		
启动模式	AMI 的启动模式。	字符串	否		有效值：legacy-bios、uefi、uefi 首选
描述	对 AMI 的描述。	字符串	否		
enaSupport	是否启用了 ENA 增强联网。	布尔值	否		
图像位置	AMI 清单的位置。	字符串	否		支持 S3 的必填项 AMIs
IMDS 支持	支 IMDSv2 撑级别。	字符串	否		有效值：v2.0
includeSnapshotTags	是否包含块储存设备映射中定义的第一个快照中的标签。	布尔值	否	FALSE	设置为 true 时，将按如下方式包含标签： • `blockDeviceMapping` 列表中`SnapshotId` 包含 a `SnapshotId` 的第一个 EBS 卷的标签将与输出 AMI 标签合并。 • 输出 AMI 标签优先于具有相同密钥的快照标签。

输入名称	说明	Type	必需	默认	约束
					• AWS 保留的标签（密钥以开头的标签aws:）会被自动排除在外。 • 如果定义了多个 EBS 卷，SnapshotId 则仅包含列表中第一个 EBS 卷中包含 a SnapshotId 的标签。
kerneLid	要使用的内核的 ID。	字符串	否		
ramdisKid	要使用的 RAM 磁盘的 ID。	字符串	否		
rootDeviceName	根设备的设备名称。	字符串	否		示例：/dev/sda1
sriovNetSupport	使用英特尔 82599 VF 接口增强联网。	字符串	否		
tpm 支持	TPM 版本支持。	字符串	否		有效值：v2.0
uefidata	Base64 编码的 UEFI 数据。	字符串	否		
虚拟化类型	虚拟化类型。	字符串	否		有效值：hvm、半虚拟化

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
imageId	已注册图像的 AMI ID。	字符串

需要 IAM 权限

您的自定义执行角色必须具有以下权限才能使用此步骤操作：

允许操作

- ec2:DescribeSnapshots
- ec2:CreateTags

示例

在工作流文档中指定步骤操作。

```
- name: RegisterNewImage
  action: RegisterImage
  onFailure: Abort
  inputs:
    architecture: "x86_64"
    bootMode: "uefi"
    blockDeviceMapping:
      - DeviceName: "/dev/sda1"
        Ebs:
          SnapshotId: "snap-1234567890abcdef0"
          VolumeSize: 100
          VolumeType: "gp3"
    rootDeviceName: "/dev/sda1"
    virtualizationType: "hvm"
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.RegisterNewImage.imageId
```

示例： SnapshotId 来自另一个步骤，生成的 AMI 中包含快照标签

```
- name: CreateSnapshot
```

```
  action: RunCommand
  onFailure: Abort
  inputs:
    instanceId: "i-1234567890abcdef0"
    documentName: "AWS-RunShellScript"
    parameters:
      commands:
        - "aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description
 'Snapshot for AMI' --query 'SnapshotId' --output text"

- name: RegisterImageFromSnapshot
  action: RegisterImage
  onFailure: Abort
  inputs:
    architecture: "x86_64"
    bootMode: "uefi"
    blockDeviceMapping:
      - DeviceName: "/dev/sda1"
        Ebs:
          SnapshotId.$: "$.stepOutputs.CreateSnapshot.output[0]"
          VolumeSize: 100
          VolumeType: "gp3"
    includeSnapshotTags: true
    rootDeviceName: "/dev/sda1"
    virtualizationType: "hvm"
```

RunCommand

此步骤操作为您的工作流运行命令文档。Image Builder 使用 Systems Manager API 中的 sendCommand 来为您运行命令文档。有关更多信息，请参阅 AWS Systems Manager Run Command。

默认超时：720 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要在其上运行命令文档的实例的 ID。	字符串	是		这必须是启动此工作流实例的工作流步骤的输出实例 ID。
documentName	要运行的 Systems Manager 命令文档的名称。	字符串	是		
参数	命令文档所需的任何参数的键值对列表。	dictionary<string, list<string>>	有条件		
documentVersion	要运行的命令文档版本。	字符串	否	$DEFAULT	

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
runCommandId	在实例上运行命令文档的 Systems Manager sendCommand 的 ID。	字符串
status	从 Systems Manager sendCommand 返回的状态。	字符串
output	从 Systems Manager sendCommand 返回的输出。	字符串列表

示例

在工作流文档中指定步骤操作。

```
- name: RunCommandDoc
  action: RunCommand
  onFailure: Abort
  inputs:
    documentName: SampleDocument
    parameters:
        osPlatform:
          - "linux"
    instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.RunCommandDoc.status
```

RunSysPrep

此步骤操作使用 Systems Manager API 中的 sendCommand 为 Windows 实例运行 AWSEC2-RunSysprep 文档，然后为快照关闭构建实例。 这些操作遵循了强化和清理图像AWS 的最佳实践。

默认超时：60 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要在其上运行 AWSEC2-RunSysprep 文档的实例的 ID。	字符串	是		这必须是启动此工作流实例的工作流步骤的输出实例 ID。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
runCommandId	在实例上运行 AWSEC2-RunSysprep 文档的 Systems Manager sendCommand 的 ID。	字符串
status	从 Systems Manager sendCommand 返回的状态。	字符串
output	从 Systems Manager sendCommand 返回的输出。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: RunSysprep
  action: RunSysPrep
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.RunSysprep.status
```

SanitizeInstance

此操作步骤会运行适用于 Linux 实例的推荐清理脚本，然后为快照关闭构建实例。清理脚本可帮助确保最终映像遵循安全最佳实践，并删除不应延续到快照中的构建构件或设置。有关脚本的更多信息，请参阅需要在构建后进行清理。此步骤操作不适用于容器映像。

Image Builder 使用 Systems Manager API 中的 sendCommand 来运行此脚本。有关更多信息，请参阅 AWS Systems Manager Run Command。

默认超时：60 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要清理的实例的 ID。	字符串	是		这必须是启动此工作流实例的工作流步骤的输出实例 ID。

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
runCommandId	在实例上运行清理脚本的 Systems Manager sendCommand 的 ID。	字符串
status	从 Systems Manager sendCommand 返回的状态。	字符串
output	从 Systems Manager sendCommand 返回的输出。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: SanitizeStep
  action: SanitizeInstance
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.SanitizeStep.status
```

TerminateInstance

此步骤操作使用作为输入传入的实例 id 终止实例。

默认超时：30 分钟

最大超时：720 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要终止的实例的 ID。	字符串	是		

输出：此步骤操作没有输出。

示例

在工作流文档中指定步骤操作。

```
- name: TerminateInstance
  action: TerminateInstance
  onFailure: Continue
  inputs:
      instanceId.$: i-1234567890abcdef0
```

WaitForAction

此步骤操作会暂停正在运行的工作流，并等待接收来自 Image Builder SendWorkflowStepAction API 操作的外部操作。此步骤使用详细信息类型将 EventBridge 事件发布到您的默认 EventBridge 事件总线EC2 Image Builder Workflow Step Waiting。如果您提供 SNS 主题 ARN，则该步骤还可以发送 SNS 通知；如果您提供 Lambda 函数名称，则该步骤还可以异步调用 Lambda 函数。

默认超时：3 天

最大超时时间：7 天

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
snsTopicArn	可选的 SNS 主题 ARN，用于在工作流步骤处于待处理状态时向其发送通知。	字符串	否		
lambdaFunctionName	Lambda 函数的可选名称或 ARN，用于在工作流程步骤处于待处理状态时异步调用。	字符串	否		
payload	JSON 字符串用作 SNS 的消息和 Lambda 的有效负载。如果提供，则默认封装自定义有效负载message/payload, used for SNS and Lambda respectively. If not provided, generates	字符串	否		必须是有效的 JSON 字符串，最大 16 KiB

输入名称	说明	Type	必需	默认	约束
	default message/payload。				

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
action	SendWorkflowStepAction API 操作返回的操作。	字符串（RESUME 或 STOP）
reason	返回操作的原因。	字符串

需要 IAM 权限

您的自定义执行角色必须具有以下权限才能使用此步骤操作：

允许操作

- `lambda:InvokeFunction`

指定资源

- `arn:aws:lambda:us-west-2:111122223333:function:function-name`
- `arn:aws:lambda:us-west-2:111122223333:function:*`

示例

在带有 SNS 通知的工作流程文档中指定步骤操作。

```
- name: SendEventAndWait
  action: WaitForAction
  onFailure: Abort
  inputs:
    snsTopicArn: arn:aws:sns:us-west-2:111122223333:ExampleTopic
```

使用 Lambda 函数调用在工作流程文档中指定步骤操作。

```
- name: SendEventAndWaitWithLambda
  action: WaitForAction
  onFailure: Abort
  inputs:
    lambdaFunctionName: ExampleFunction
    payload: |
      {
        "imageId": "{{ $.stepOutputs.CreateImageFromInstance.imageId }}",
        "region": "us-west-2"
      }
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.SendEventAndWait.reason
```

WaitForSSMAgent

此步骤操作等待 EC2 实例在预期的无响应时间 AWS Systems Manager 之后变得可控。它对于存在已知实例中断的工作流程（例如系统重启、操作系统升级或暂时断开实例与 SSM 连接的特定于平台的操作）特别有用。Image Builder 会监控该实例，直到它恢复 SSM 连接或超时。

默认超时：60 分钟

最大超时时间：180 分钟

回滚：此步骤操作没有回滚。

输入：下表包含此步骤操作支持的输入。

输入名称	说明	Type	必需	默认	约束
instanceId	要监控 SSM 连接的实例的 ID。	字符串	是		必须是有效的 EC2 实例 ID

输出：下表包含此步骤操作的输出。

输出名称	说明	Type
status	SSM 代理的连接状态。	字符串

示例

在工作流文档中指定步骤操作。

```
- name: WaitForInstanceAfterReboot
  action: WaitForSSMAgent
  onFailure: Abort
  timeoutInSeconds: 900
  inputs:
    instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

在工作流文档中使用步骤操作值的输出。

```
$.stepOutputs.WaitForInstanceAfterReboot.Status
```

在工作流文档中使用动态变量

您可以在工作流文档中使用动态变量来表示映像创建过程中在运行时会发生变化的值。动态变量的字符串插值允许您在结构化内容（例如 JSON 字符串）中嵌入 JSONPath 表达式。当您需要将复杂负载中的运行时值传递给步进操作（如或）时，这特别有用。`ExecuteStateMachine` `WaitForAction`

要对动态变量使用字符串插值，请在字符串内容中将 JSONPath 表达式用双大括号`"{{...}}"`括起来。只有用双花括号括起来的 JSONPath 表达式才会被当作变量处理。任何未用双大括号括起的 JSONPath 表达式都被视为文字字符串内容。

JSONPath 动态工作流程变量语法

```
$.<document structure>.[<step name>.]<variable name>
```

动态变量值表示为具有唯一标识目标变量的结构节点的 JSONPath 选择器。根 ($) 之后的第一个节点是指工作流文档结构，例如 `stepOutputs`，如果是 Image Builder 系统变量，则为 `imageBuilder`。以下列表包含支持 JSONPath 的工作流文档结构节点。

文档结构节点

- 参数 – 工作流参数
- stepOutputs – 同一工作流文档中某个步骤的输出
- workflowOutputs – 已运行的工作流文档的输出
- imagebuilder – Image Builder 系统变量

`parameters` 和 `stepOutputs` 文档结构节点包括步骤名称的可选节点。这样有助于确保所有步骤中变量名称的唯一性。

中的最后一个节点 JSONPath 是目标变量的名称，例如`instanceId`。

每个步骤都可以使用这些 JSONPath 动态变量引用任何先前步骤操作的输出。这也称为链接或引用。要引用先前步骤操作的输出，可以使用以下动态变量。

```
$.stepOutputs.step-name.output-name
```

> **Important**
>
> 当输入参数引用动态变量时，必须将链接指示符 (`.$`) 附加到参数名称的末尾。

示例 1：输入参数链接指示器

以下示例显示了一个输入参数，该参数在运行时使用字符串插值来解析参数值中的动态变量。

```
  - name: ApplyTestComponents
    action: ExecuteComponents
    onFailure: Abort
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"
```

示例 2：动态变量中的字符串插值

以下示例演示了动态变量如何在运行时使用字符串插值来确定值。

```
- name: ValidateImageConfiguration
  action: ExecuteStateMachine
  inputs:
    stateMachineArn: arn:aws:states:us-east-1:111122223333:stateMachine:ImageValidation
```

```
    input: |
      {
        "imageId": "{{ $.stepOutputs.CreateImageFromInstance.imageId }}",
        "region": "us-east-1",
        "buildDate": "{{ $.imagebuilder.dateTime }}",
        "instanceType": "{{ $.stepOutputs.LaunchStep.instanceType }}"
      }
```

在此示例中，用双大括号包裹的 JSONPath 表达式在运行时被解析：

- `{{ $.stepOutputs.CreateImageFromInstance.imageId }}`-解析为步骤中的实际图片 ID CreateImageFromInstance
- `{{ $.imagebuilder.dateTime }}`-解析为当前构建时间戳。有关您可以使用使用 Image Builder 系统变量的 Image Builder 系统变量的列表，请参阅。
- `{{ $.stepOutputs.LaunchStep.instanceType }}`-解析为中使用的实例类型 LaunchStep

像这样的字面字符串`"region": "us-east-1"`保持不变。

> **Note**
>
> 字符串插值适用于工作流程文档中的任何字符串内容，包括使用 YAML pipe () 运算符的多行字符串。| 大括号要求充当一种转义机制，可以清楚地区分JSONPath 变量和文字文本内容。

使用 Image Builder 系统变量

Image Builder 提供了以下可在工作流文档中使用的系统变量：

变量名称	说明	Type	示例值
cloudWatchLog群组	输出 CloudWatch 日志的日志组的名称。 格式：/aws/imagebuilder/*<recipe-name>*	字符串	/aws/imagebuilder/*sampleImageRecipe*

变量名称	说明	Type	示例值
cloudWatchLog直播	输出 CloudWatch 日志的日志流的名称。	字符串	*1.0.0/1*
collectImageMetadata	指示 Image Builder 是否收集实例元数据的设置。	布尔值	`true` | `false`
collectImageScan调查结果	设置的当前值，该设置允许 Image Builder 收集映像扫描调查发现。	布尔值	`true` | `false`
imageBuildNumber	映像的构建版本号。	整数	*1*
imageId	基础映像的 AMI ID。	字符串	*ami-12345 67890abcdef1*
imageName	映像名称。	字符串	*sampleImage*
imageType	映像输出类型。	字符串	`AMI` | `Docker`
imageVersionNumber	映像版本号。	字符串	*1.0.0*
instanceProfileName	Image Builder 用于启动构建和测试实例的实例配置文件角色的名称。	字符串	*SampleIma geBuilder InstanceP rofileRole*
platform	所构建映像的操作系统平台。	字符串	`Linux` | `Windows` | `MacOS`

变量名称	说明	Type	示例值
s3Logs	一个 JSON 对象，包含 Image Builder 写入的 S3 日志的配置。	JSON 对象	{'s3Logs': {'s3 BucketName ':*sample-bucket* ', '3 ':KeyPrefix'}} *ib-logs*
securityGroups	适用于构建和测试实例的安全组 IDs 。	列表 [字符串]	*[sg-12345 67890abcd ef1, sg-111122 22333344445]*
sourceImageARN	工作流用于构建和测试阶段的 Image Builder 映像资源的 Amazon 资源名称（ARN）。	字符串	arn: aws: imagebuil der::: image//*us-east-1 111122223 333 sampleIma ge 1.0.0/1*
subnetId	要在其中启动构建和测试实例的子网 ID。	字符串	*subnet-12 34567890a bcdef1*
terminateInstanceOn 失败	设置的当前值，可指示 Image Builder 在发生故障时终止实例或保留实例以进行故障排除。	布尔值	`true` | `false`
workflowPhase	为执行工作流而运行的当前阶段。	字符串	`Build` | `Test`

变量名称	说明	Type	示例值
workingDirectory	工作目录的路径。	字符串	/tmp

在工作流步骤中使用条件语句

条件语句以 if 语句文档属性开头。if 语句的最终目的为确定是运行还是跳过该步骤操作。如果 if 语句解析为 true，则该步骤操作将运行。如果解析为 false，Image Builder 将跳过该步骤操作并在日志中记录 SKIPPED 的步骤状态。

if 语句支持分支语句（and、or）和条件修饰符 (not)。其还支持以下比较运算符，这些运算符根据所比较的数据类型（字符串或数字）执行值比较（等于、小于、大于）。

支持的比较运算符

- booleanEquals
- numberEquals
- numberGreaterThan
- numberGreaterThanEquals
- numberLessThan
- numberLessThanEquals
- stringEquals

分支语句和条件修饰符的规则

以下规则适用于分支语句（and、or）和条件修饰符 (not)。

- 分支语句和条件修饰符必须单独出现在某一行上。
- 分支语句和条件修饰符必须遵循级别规则。
  - 父级别只能有一个语句。
  - 每个子分支或修饰符都会启动一个新级别。

  有关级别的更多信息，请参阅 条件语句中的嵌套级别。
- 每个分支语句必须至少包含一个子条件语句，但不能超过十个。

- 条件修饰符仅对一个子条件语句执行操作。

条件语句中的嵌套级别

条件语句在自身部分中的多个级别上执行操作。例如，if 语句属性在工作流文档中与步骤名称和操作显示在同一级别。这是条件语句的基础。

您可以指定最多四个级别的条件语句，但父级别只能显示一个语句。所有其他分支语句、条件修饰符或条件运算符都从此处缩进，每个级别缩进一个。

以下概述显示了条件语句的最大嵌套级别数。

```
base:
  parent:
    - child (level 2)
        - child (level 3)
            child (level 4)
```

if 属性

if 属性将条件语句指定为文档属性。这是零级。

父级别

这是条件语句的第一级嵌套。此级别只能有一个语句。如果不需要分支或修饰符，则可以为没有子语句的条件运算符。除条件运算符外，此级别不使用连接号表示法。

子级别

二级到四级被视为子级别。子语句可以包括分支语句、条件修饰符或条件运算符。

示例：嵌套级别

以下示例演示了条件语句的最大级别数。

```
if:
  and:                              #first level
    - stringEquals: 'my_string'     #second level
      value: 'my_string'
    - and:                          #also second level
        - numberEquals: '1'         #third level
          value: 1
```

```
        - not:                      #also third level
            stringEquals: 'second_string'      #fourth level
            value: "diff_string"
```

嵌套规则

- 子级的每个分支或修饰符都会启动一个新级别。
- 每个级别都可缩进。
- 最多可以有四个级别，包括父级别的一个语句、修饰符或运算符，以及最多三个其他级别。

条件语句示例

这组示例演示了条件语句的各个方面。

分支：and

and 分支语句对作为分支子级的表达式列表执行操作，所有这些表达式的计算结果都必须为 `true`。Image Builder 按照表达式在列表中出现的顺序计算表达式。如果任何表达式的计算结果为 `false`，则停止处理，该分支将视为 `false`。

以下示例的计算结果为 `true`，因为两个表达式的计算结果均为 `true`。

```
if:
  and:
    - stringEquals: 'test_string'
      value: 'test_string'
    - numberEquals: 1
      value: 1
```

分支：or

or 分支语句对作为该分支子级的表达式列表执行操作，其中至少有一个表达式的计算结果必须为 `true`。Image Builder 按照表达式在列表中出现的顺序计算表达式。如果任何表达式的计算结果为 `true`，则停止处理，该分支将视为 `true`。

即使第一个表达式为 `false`，以下示例的计算结果仍为 `true`。

```
if:
  or:
    - stringEquals: 'test_string'
```

```
      value: 'test_string_not_equal'
    - numberEquals: 1
      value: 1
```

条件修饰符：not

not 条件修饰符否定作为分支子级的条件语句。

当 not 修饰符否定 stringEquals 条件语句时，以下示例的计算结果为 true。

```
if:
  not:
    - stringEquals: 'test_string'
      value: 'test_string_not_equal'
```

条件语句：booleanEquals

booleanEquals 比较运算符对布尔值进行比较，如果布尔值完全匹配，则返回 true。

以下示例可确定 collectImageScanFindings 是否启用。

```
if:
  - booleanEquals: true
    value: '$.imagebuilder.collectImageScanFindings'
```

条件语句：stringEquals

stringEquals 比较运算符对两个字符串进行比较，如果两个字符串完全匹配，则返回 true。如果其中一个值不是字符串，则 Image Builder 会在比较之前将其转换为字符串。

以下示例对平台系统变量进行了比较，以确定工作流是否在 Linux 平台上运行。

```
if:
  - stringEquals: 'Linux'
    value: '$.imagebuilder.Platform'
```

条件语句：numberEquals

numberEquals 比较运算符对两个数字进行比较，如果两个数字相等，则返回 true。要比较的数字必须为以下格式之一。

- 整数

- 浮点型
- 与以下正则表达式模式相匹配的字符串：^-?[0-9]+(\.)?[0-9]+$。

以下示例比较的计算结果均为 true。

```
if:
  # Value provider as a number
  numberEquals: 1
  value: '1'

  # Comparison value provided as a string
  numberEquals: '1'
  value: 1

  # Value provided as a string
  numberEquals: 1
  value: '1'

  # Floats are supported
  numberEquals: 5.0
  value: 5.0

  # Negative values are supported
  numberEquals: -1
  value: -1
```

通过可重复的管道流程在 Image Builder 中管理自定义映像创建

Image Builder 镜像管道为创建和维护自定义镜像 AMIs 和容器镜像提供了一个自动化框架。管道提供以下功能：

- 组装基础映像，构建和测试用于镜像自定义、基础架构配置和分发设置的组件。
- 使用控制台向导中的 `Schedule builder` 方便地安排自动维护流程，或输入 cron 表达式以对映像进行定期更新。
- 为基础映像和组件启用更改检测，以便在没有更改时自动跳过计划构建。
- 通过 Amazon 启用基于规则的自动化。 EventBridge

> **Note**
>
> 有关使用 EventBridge API 查看或更改规则的更多信息，请参阅 Amazon EventBridge API 参考。有关使用 EventBridgeevents命令查看或更改规则 AWS CLI 的更多信息，请参阅《AWS CLI 命令参考》中的事件。

主题

为图像管道配置管道执行设置

您可以从以下选项中进行选择来安排管道的执行：

日程生成器

使用计划生成器配置自动重复的管道执行。您可以定义管道运行的时间和频率（日期、时间和频率）。默认计划为每周一次，基于创建计划的日期和时间 (UTC)。

Cron 表达式

使用指定时间表的 cron 表达式自动运行管道。有关 Image Builder 使用的 cron 语法的更多信息，请参阅在 Image Builder 中使用 cron 表达式。

手动

管道未按计划运行。在控制台中，从 “操作” 菜单中选择 “运行管道” 以运行管道。从那 AWS CLI 里，你可以跑`start-image-pipeline-execution`了。

依赖项设置

对于计划生成，您可以选择是始终按计划运行还是跳过管道执行，除非有依赖项更新，例如更改基础映像或配方中使用的组件。

自动禁用失败的管道

对于按计划运行的图像管道，在 Image Builder 自动禁用管道之前，您可以配置允许的最大连续失败次数（最多`10`）。

自动禁用设置

Image Builder 会跟踪计划管道执行的连续失败次数，并在每次按计划运行时执行以下操作之一：

- 如果管道执行成功，则连续失败的次数将重置为零。
- 如果管道执行失败，Image Builder 会增加连续失败的次数。如果失败次数超过中定义的限制`AutoDisablePolicy`，Image Builder 将禁用管道。

在以下条件下，连续的失败计数也会重置为零：

- 管道手动运行并成功运行。
- 管道配置已更新。

如果管道手动运行但失败，则计数保持不变。下一次计划运行继续从之前停下来的地方递增。

配置管道日志

创建或更新映像管道时，您可以为映像构建和管道 CloudWatch 日志配置自定义日志组。确保您的自定义管道执行角色具有以下权限来创建和访问日志组资源。

- 日志：CreateLogGroup
- 日志：CreateLogStream
- 日志：PutLogEvents

自定义日志组

要使用自定义日志组进行映像构建或管道执行，请先在 “日志” 中创建 CloudWatch 日志组。有关更多信息，请参阅 Amazon 日志用户指南中的创建 CloudWatch 日志组。有关日志组命名要求的更多指导，请参阅 Amazon CloudWatch 日志 API 参考CreateLogGroup中的。

Console

在 “高级设置” 下的 “日志配置” 部分中为您的管道指定图像日志组或管道日志组。

CLI

如果您使用 JSON `logging-configuration` 对象进行配置，请在对象中指定以下字段：

- `imageLogGroupName`
- `pipelineLogGroupName`

要直接在命令行中指定所有参数，请参阅《AWS CLI 命令参考》create-image-pipeline中的。

如果您未指定自定义日志组，Image Builder 将使用以下默认日志组：

镜像构建日志

Image Builder 将构建日志写入以下 Image Builder CloudWatch 日志组并进行直播：

LogGroup: `/aws/imagebuilder/`*`ImageName`*

LogStream (x.x.x/x): *`ImageVersion/ImageBuildVersion`*

管道执行日志

Image Builder 将管道执行日志写入以下 Image Builder CloudWatch 日志组并进行流式传输：

LogGroup: /aws/imagebuilder/pipeline/*pipeline-name*

LogStream:*2025/09/01*（YYYY/MM/DD格式为管道执行日期）

每条管道日志都会附加到当天的直播中。

手动运行镜像管道

如果您为管道选择了手动计划选项，则只有在您手动启动构建时它才会运行。如果您选择了自动计划选项之一，则还可以在定期计划运行之间手动运行管道。例如，如果您的管道通常每月运行一次，但需要在上一次运行两周后对其中一个组件进行更新，则可以选择手动运行管道。

Console

要从 Image Builder 控制台的管道详细信息页面运行管道，请从页面顶部的操作菜单中选择运行管道。会在页面顶部显示一条状态消息，提示您管道已启动或是否存在错误。

1. 在管道详细信息页面的左上角，选择操作，然后选择编辑管道。
2. 您可以在输出映像选项卡的状态列中查看管道的当前状态。

AWS CLI

以下示例说明了如何在 AWS CLI 中使用 start-image-pipeline-execution 命令手动启动镜像管道。运行此命令时，管道会构建并分配新映像。

```
aws imagebuilder start-image-pipeline-execution --image-pipeline-arn
 arn:aws:imagebuilder:us-west-2:111122223333: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。

在 Image Builder 中使用 cron 表达式

使用 EC2 Image Builder 的 cron 表达式来设置时间窗口，使用适用于管道基础图像和组件的更新来刷新图像。管道刷新的时间窗口从您在 cron 表达式中设置的时间开始。您可以在 cron 表达式中将时间设置为分钟。您的管道构建可以在开始时间或之后运行。

有时可能需要几秒钟或长达一分钟的时间构建才能开始运行。

> **Note**
>
> 默认情况下，Cron 表达式使用通用协调时间 (UTC) 时区，或者您也可以自己指定时区。有关 UTC 时间的更多信息以及要查找您所在时区的偏移量，请参阅时区缩写-全球列表。

镜像生成器中 Cron 表达式支持的值

EC2 Image Builder 使用由六个必填字段组成的 cron 格式。每个字段都用一个空格与其他字段隔开，没有前导或尾随空格：

<Minute> <Hour> <Day> <Month> <Day of the week> <Year>

下表列出了必需的 cron 条目支持的值。

Cron 表达式支持的值

字段	值	通配符
分钟	0-59	, - * /
小时	0-23	, - * /
天	1-31	, - * ? / L W
Month	1-12 或 jan-dec	, - * /
一星期中的日子。	1-7 或 sun-sat	, - * ? L #
Year	1970-2199	, - * /

通配符

下表描述 Image Builder 如何在 cron 表达式中使用通配符。请记住，在您指定的构建时间之后，最多可能需要一分钟才能开始构建。

Cron 表达式支持的通配符

通配符	描述
,	, (逗号) 通配符包含其他值。在字段中，`jan,feb,mar` 将包含 January、February 和 March。
-	- (破折号) 通配符用于指定范围。在“日”字段中，`1-15`1-15 将包含指定月份的 1 - 15 日。
*	* (星号) 通配符包含该字段的所有有效值。
?	? (问号) 通配符用于指定字段值取决于其他设置。对于日期和 Day-of-week字段，当指定一个或包含所有可能的值 (*) 时，另一个必须是 a ?。不能同时指定两者。例如，如果您在 “日” 字段7中输入 a (在当月的第七天运行构建) ，则该 Day-of-week位置必须包含?。
/	/ (正斜杠) 通配符用于指定增量。例如，如果您希望每隔一天运行构建，请在“日”字段中输入 `*/2`。
L	任一“日”字段中的 L 通配符指定最后一天：“28-31” 表示月份的最后一天，具体取决于月份，或者“星期日”表示一周中的最后一天。
W	Day-of-month 字段中的 W 通配符用于指定工作日。在该 Day-of-month字段中，如果您在之前输入一个数字`W`，则表示您要将最接近该天的工作日作为目标。例如，如果您指定 `3W`，则希望您的构建在最接近该月第三天的工作日运行。
#	# (hash) 仅允许用于“星期几”字段，并且后面必须有一个介于 1 到 5 之间的数字。该数字指定了给定月份中哪几周需要运行构建。例如，如果您希望在每个月的第二个星期五运行构建，请使用 `fri#2` 用于“星期几”字段。

限制

- 您无法在同一 cron 表达式中为 Day-of-month 和 Day-of-week 字段同时指定值。如果您在其中一个字段中指定了值*（或一个 *），则必须在另一个字段中使用 ??。
- 不支持产生的速率快于一分钟的 Cron 表达式。

Image Builder 中的 cron 表达式示例

在 Image Builder 控制台中输入的 Cron 表达式与 API 或 CLI 中输入的 Cron 表达式不同。要查看示例，请选择适用于您的选项卡。

Image Builder console

以下示例显示了可以在控制台中输入构建计划的 cron 表达式。UTC 时间使用 24 小时制指定。

每天上午 10:00 (UTC) 运行

```
0 10 * * ? *
```

每天上下午 12:15 (UTC) 运行

```
15 12 * * ? *
```

UTC 时间每天午夜运行

```
0 0 * * ? *
```

每个工作日上午 10:00 (UTC) 运行

```
0 10 ? * 2-6 *
```

每个工作日晚上 6 点 (UTC) 运行

```
0 18 ? * mon-fri *
```

每月第 1 天上午 8:00 (UTC) 运行

```
0 8 1 * ? *
```

每个月的第二个星期二晚上 10:30（UTC）运行

```
30 22 ? * tue#2 *
```

> **Tip**
>
> 如果您不希望管道作业在运行时延长到第二天，请确保在指定开始时间时将构建时间考虑在内。

API/CLI

以下示例显示了可以使用 CLI 指令或 API 请求输入构建计划的 cron 表达式。仅显示 cron 表达式。

每天上午 10:00 (UTC) 运行

```
cron(0 10 * * ? *)
```

每天上下午 12:15 (UTC) 运行

```
cron(15 12 * * ? *)
```

UTC 时间每天午夜运行

```
cron(0 0 * * ? *)
```

每个工作日上午 10:00 (UTC) 运行

```
cron(0 10 ? * 2-6 *)
```

每个工作日晚上 6 点 (UTC) 运行

```
cron(0 18 ? * mon-fri *)
```

每月第 1 天上午 8:00 (UTC) 运行

```
cron(0 8 1 * ? *)
```

每个月的第二个星期二晚上 10:30（UTC）运行

```
cron(30 22 ? * tue#2 *)
```

> **Tip**
>
> 如果您不希望管道作业在运行时延长到第二天，请确保在指定开始时间时将构建时间考虑在内。

Image Builder 中的 rate 表达式

Rate 表达式在创建计划事件规则时启动，然后按照其定义的计划运行。

Rate 表达式有两个必需字段。这些字段用空格分隔。

语法

```
rate(value unit)
```

值

一个正数。

unit

时间单位。需要不同的单位，例如，对于值 1 为 `minute`；对于大于 1 的值 1 为 `minutes`。

有效值：minute | minutes | hour | hours | day | days

限制

如果值等于 1，则单位必须为单数。同样，对于大于 1 的值，单位必须为复数。例如，`rate(1 hours)` 和 `rate(5 hour)` 无效，但 `rate(1 hour)` 和 `rate(5 hours)` 有效。

列出并查看管道详情

本部分描述查找 EC2 Image Builder 映像管道的信息和查看详细信息的多种方式。

管道详细信息

- 列出来自的图像管道 AWS CLI
- 从中获取图像管道的详细信息 AWS CLI

列出来自的图像管道 AWS CLI

以下示例说明如何使用中的list-image-pipelines命令列 AWS CLI 出所有图像管道。

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

从中获取图像管道的详细信息 AWS CLI

以下示例说明如何使用中的get-image-pipeline命令通过 ARN 获取有关图像管道的详细信息。 AWS CLI

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

创建和更新 AMI 映像管道

您可以通过映像生成器控制台、Image Builder API SDKs 或者，设置、配置和管理 AMI 图像管道 AWS CLI。在控制台中，您可以使用创建映像管道控制台向导来指导您完成以下步骤。

- 指定管道详细信息，例如名称、描述和资源标签。
- 配置管道计划和日志记录默认值。对于定时管道执行，您可以设置在 Image Builder 禁用管道之前允许的连续失败次数。
- 选择一个 AMI 图像配方，其中包含来自快速入门的 Amazon 托管映像的基本映像、您创建或与您共享的图像或您通过订阅的 AWS Marketplace图像。该配方还包括在 Image Builder 用于构建映像的 EC2 实例上执行以下任务的组件：
  - 添加和删除软件
  - 自定义设置和脚本
  - 运行选定的测试
- 指定工作流以配置管道运行的映像构建和测试步骤。
- 使用默认设置或您自己配置的设置为管道定义基础设施配置。该配置包括用于映像的实例类型和密钥对、安全和网络设置、日志存储和故障排除设置以及 SNS 通知。

  此为可选步骤。如果您没有自己定义配置，Image Builder 会使用默认设置来配置基础设施。
- 定义分配设置，将您的映像发送到目标 AWS 地区和账户。您可以指定用于加密的 KMS 密钥、配置 AMI 共享或许可证配置，或者 AMIs 为您分发的配置启动模板。

  此为可选步骤。如果您没有自己定义配置，Image Builder 会使用输出 AMI 的默认命名，并将 AMI 分配到源区域。源区域是您运行管道的区域。

有关使用默认值创建映像管道控制台向导的更多信息和 step-by-step教程（如有），请参阅教程：通过 Image Builder 控制台向导使用输出 AMI 创建映像管道。

内容

从创建 AMI 映像管道 AWS CLI

要从中创建映像管道 AWS CLI，请使用适用于您的管道的配置选项运行create-image-pipeline命令。您可以选择创建包含所有工作流配置的 JSON 文件，或者在运行时指定配置。本节使用 JSON 配置文件方法来简化命令。

您的管道构建新映像以合并基础映像和组件中所有待处理更新的频率取决于您所配置的 `schedule`。每个 `schedule` 都具有以下属性：

- `scheduleExpression`— 设置管道运行的时间计划，以评估 `pipelineExecutionStartCondition` 并确定是否应开始构建。该计划是使用 cron 表达式配置的。有关如何在 Image Builder 中设置 cron 表达式格式的更多信息，请参阅 在 Image Builder 中使用 cron 表达式。
- `pipelineExecutionStartCondition`— 确定您的管道是否应该开始构建。有效值包括：
  - `EXPRESSION_MATCH_ONLY`— 每次 cron 表达式与当前时间匹配时，Pipeline 都会生成一个新映像。
  - `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`— 除非您的基础映像或组件有待处理的更改，否则您的管道不会启动新的映像构建。

在中运行create-image-pipeline命令时 AWS CLI，许多配置资源都是可选的。但是，有些资源有条件性要求，具体取决于管道创建的映像类型。AMI 图像管道需要以下资源标识符：

- 映像配方 ARN
- 基础设施配置 ARN

示例：创建 Windows 2019 镜像

此示例配置了一个定于每周星期日运行一次的管道。第一步中显示的配置文件使用映像配方、基础架构和分发配置的现有资源以及其他设置来创建 Windows 2019 映像。

1. 创建配置文件（可选）

   此示例使用名create-image-pipeline.json为的配置文件在一个位置配置设置。或者，您可以在运行命令时使用命令行选项来指定配置文件中此处显示的所有详细信息。

```
{
 "name": "ExampleWindows2019Pipeline",
 "description": "Builds Windows 2019 Images",
 "enhancedImageMetadataEnabled": true,
 "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2020.12.03",
 "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"
}
```

   > **Note**
   >
   > - JSON 文件路径开头必须包含 `file://` 符号。
   > - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

2. 运行命令创建镜像

   此示例使用在第一步中创建的配置文件作为create-image-pipeline命令的输入。或者，您可以在运行命令时直接为管道指定设置和资源。有关更多信息，请参阅 AWS CLI 参考中的 create-image-pipeline。

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

通过控制台更新 AMI 映像管道

为 AMI 映像创建 Image Builder 映像管道后，您可以从 Image Builder 控制台更改基础设施配置和分配设置。

要使用新的映像配方更新映像管道，必须使用 AWS CLI。有关更多信息，请参阅本指南中的从更新 AMI 图像管道 AWS CLI。

选择现有的 Image Builder 管道

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 要查看在您的账户下创建的映像管道列表，请从导航窗格中选择映像管道。

> **Note**
>
> 映像管道列表包括管道创建的输出映像类型的指示器，即 AMI 或 Docker。

3. 要查看详细信息或编辑管道，请选择管道名称链接。这将打开管道的详细视图。

> **Note**
>
> 您也可以选中管道名称旁边的框，然后选择查看详细信息。

管道详细信息

管道详细信息页面包括以下几节：

摘要

页面顶部的一节汇总了管道的关键详细信息，这些信息在打开任何详细信息选项卡时都可见。本节中显示的详细信息只能在其各自的详细信息选项卡上进行编辑。

详细信息 选项卡

- 输出映像-显示管道生成的输出映像。

- 映像配方-显示配方详细信息。配方一经创建即无法编辑。您必须在 Image Builder 控制台的映像配方页面或使用 AWS CLI中的 Image Builder 命令创建新版本配方。有关更多信息，请参阅 在 Image Builder 中管理配方。
- 基础设施配置-显示用于配置构建管道基础设施的可编辑信息。
- 分配设置-显示 AMI 分配的可编辑信息。
- EventBridge r ules — 对于选定的事件总线，显示针对当前管道的 EventBridge 规则。包括链接到 EventBridge 控制台的 “创建事件总线” 和 “创建规则” 操作。有关此标签页的更多信息，请参阅 使用 EventBridge 规则。

编辑管道的基础设施配置

基础设施配置包括以下详细信息，您可以在创建管道后对其进行编辑：

- 基础设施配置的描述。
- 将 IAM 角色 与实例配置文件关联。
- AWS 基础设施，包括实例类型和通知的 SNS 主题。
- VPC、子网和安全组。
- 故障排除设置，包括失败时终止实例、用于连接的密钥对以及用于存储实例日志的可选 S3 存储桶位置。

要从管道详细信息页面编辑基础设施配置，请执行以下步骤：

1. 选择基础设施配置选项卡。
2. 从配置详细信息面板的右上角选择编辑。
3. 当您准备好保存对基础设施配置所做的更新时，请选择保存更改。

编辑管道的分配设置

分配设置包括以下详细信息，您可以在创建管道后对其进行编辑：

- 此分发配置的描述。
- 您分配映像的区域的区域设置。区域 1 默认为您创建管道的区域。您可以使用添加区域按钮添加要分配的区域，也可以删除除区域 1 之外的所有区域。

  区域设置包括：

- 目标区域
- 输出 AMI 名称
- 启动权限以及与之共享权限的账户
- 关联许可证（关联许可证配置）

> **Note**
>
> License Manager 设置不会在您的账户中必须启用的 AWS 区域之间复制，例如，在ap-east-1（香港）和me-south-1（巴林）地区之间。

要从管道详细信息页面编辑您的分配设置，请按照以下步骤操作：

1. 选择分配设置选项卡。
2. 从分配详细信息面板的右上角选择编辑。
3. 准备好保存更新后，选择保存更改。

编辑管道的构建计划

编辑管道页面包含以下详细信息，您可以在创建管道后对其进行编辑：

- 对您管道的描述。
- 增强型元数据收集。默认情况下，此选项处于打开状态。要将其关闭，请清除启用增强型元数据收集复选框。
- 您管道的构建计划。您可以在此更改您的计划选项和所有设置。

要从管道详细信息页面编辑您的管道，请按照以下步骤操作：

1. 在管道详细信息页面的右上角，选择操作，然后选择编辑管道。
2. 准备好保存更新后，选择保存更改。

> **Note**
>
> 有关使用 cron 表达式计划构建的更多信息，请参阅 在 Image Builder 中使用 cron 表达式。

从更新 AMI 图像管道 AWS CLI

您可以使用 JSON 文件作为 AWS CLI中的 update-image-pipeline 命令的输入来更新 AMI 映像管道。要配置 JSON 文件，您必须有 Amazon 资源名称 (ARNs) 才能引用以下现有资源：

- 要更新的映像管道
- 映像配方
- 基础设施配置
- 分配设置

您可以使用中的update-image-pipeline命令更新 AMI 映像管道， AWS CLI 如下所示：

> **Note**
>
> UpdateImagePipeline 不支持对管道进行选择性更新。您必须在更新请求中指定所有必需的属性，而不仅仅是已更改的属性。

1. 创建 CLI 输入 JSON 文件

   使用您常用的文件编辑工具创建 JSON 文件，其中包含以下密钥以及对您的环境有效的值。此示例使用名为 `create-component.json` 的文件：

```
 {
 "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"
}
```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

2. 使用创建的文件作为输入，运行以下命令。

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

创建和更新容器映像管道

您可以使用 Image Builder 控制台、Image Builder API 或 AWS CLI中的 imagebuilder 命令来设置、配置和管理容器映像管道。创建映像管道控制台向导提供了起始构件，并指导您完成以下步骤：

- 从快速入门托管映像、Amazon ECR 或 Docker Hub 存储库中选择基础映像。
- 添加和删除软件
- 自定义设置和脚本
- 运行选定的测试
- 使用预先配置的构建时变量创建 DockerFile。
- 将图像分发到 AWS 区域

有关使用创建映像管道控制台向导的更多信息和 step-by-step教程，请参阅教程：通过 Image Builder 控制台向导使用输出 Docker 容器映像创建映像管道。

内容

- 从中更新容器镜像管道 AWS CLI

从中创建容器映像管道 AWS CLI

要从中创建映像管道 AWS CLI，请使用适用于您的管道的配置选项运行create-image-pipeline命令。您可以选择创建包含所有工作流配置的 JSON 文件，或者在运行时指定配置。本节使用 JSON 配置文件方法来简化命令。

您的管道构建新映像以合并基础映像和组件中所有待处理更新的频率取决于您所配置的 `schedule`。每个 `schedule` 都具有以下属性：

- `scheduleExpression`— 设置管道运行的时间计划，以评估 `pipelineExecutionStartCondition` 并确定是否应开始构建。该计划是使用 cron 表达式配置的。有关如何在 Image Builder 中设置 cron 表达式格式的更多信息，请参阅 在 Image Builder 中使用 cron 表达式。
- `pipelineExecutionStartCondition`— 确定您的管道是否应该开始构建。有效值包括：
  - `EXPRESSION_MATCH_ONLY`— 每次 cron 表达式与当前时间匹配时，Pipeline 都会生成一个新映像。
  - `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`— 除非您的基础映像或组件有待处理的更改，否则您的管道不会启动新的映像构建。

在中运行create-image-pipeline命令时 AWS CLI，许多配置资源都是可选的。但是，有些资源有条件性要求，具体取决于管道创建的映像类型。容器映像管道需要以下资源：

- 容器配方 ARN
- 基础设施配置 ARN

如果您在运行 create-image-pipeline 命令时未包含分配配置资源，则输出映像将存储在您运行该命令的区域的容器配方中指定为目标存储库的 ECR 存储库中。如果您为管道添加了分配配置资源，则将使用您在分配中为第一个区域指定的目标存储库。

1. 创建 CLI 输入 JSON 文件

   使用您常用的文件编辑工具创建 JSON 文件，其中包含以下密钥以及对您的环境有效的值。此示例使用名为 `create-image-pipeline.json` 的文件：

   ```
   {
   ```

```
 "name": "MyWindows2019Pipeline",
 "description": "Builds Windows 2019 Images",
 "enhancedImageMetadataEnabled": true,
 "containerRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:container-
recipe/my-example-recipe/2020.12.03",
 "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"
}
```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

2. 使用创建的文件作为输入，运行以下命令。

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

通过控制台更新容器映像管道

为 Docker 映像创建 Image Builder 容器映像管道后，您可以从 Image Builder 控制台更改基础设施配置和分配设置。

要使用新的容器配方更新容器映像管道，必须使用 AWS CLI。有关更多信息，请参阅本指南中的从中更新容器镜像管道 AWS CLI。

选择现有的 Image Builder Docker 映像管道

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 要查看在您的账户下创建的映像管道列表，请从导航窗格中选择映像管道。

> **Note**
>
> 映像管道列表包括管道创建的输出映像类型的指示器，即 AMI 或 Docker。

3. 要查看详细信息或编辑管道，请选择管道名称链接。这将打开管道的详细视图。

> **Note**
>
> 您也可以选中管道名称旁边的框，然后选择查看详细信息。

管道详细信息

EC2 Image Builder 管道详细信息页面包括以下几节：

摘要

页面顶部的一节汇总了管道的关键详细信息，这些信息在打开任何详细信息选项卡时都可见。本节中显示的详细信息只能在其各自的详细信息选项卡上进行编辑。

详细信息 选项卡

- 输出映像-显示管道生成的输出映像。
- 容器配方-显示配方详细信息。配方一经创建即无法编辑。您必须从容器配方页面创建新版本配方。有关更多信息，请参阅 创建新版本的容器配方。
- 基础设施配置-显示用于配置构建管道基础设施的可编辑信息。
- 分配设置-显示 Docker 映像分配的可编辑信息。
- EventBridge r ules — 对于选定的事件总线，显示针对当前管道的 EventBridge 规则。包括链接到 EventBridge 控制台的 “创建事件总线” 和 “创建规则” 操作。有关此标签页的更多信息，请参阅 使用 EventBridge 规则。

编辑管道的基础设施配置

基础设施配置包括以下详细信息，您可以在创建管道后对其进行编辑：

- 基础设施配置的描述。
- 将 IAM 角色 与实例配置文件关联。
- AWS 基础设施，包括实例类型和通知的 SNS 主题。
- VPC、子网和安全组。
- 故障排除设置，包括失败时终止实例、用于连接的密钥对以及用于存储实例日志的可选 S3 存储桶位置。

要从管道详细信息页面编辑基础设施配置，请执行以下步骤：

1. 选择基础设施配置选项卡。
2. 从配置详细信息面板的右上角选择编辑。
3. 当您准备好保存对基础设施配置所做的更新时，请选择保存更改。

编辑管道的分配设置

分配设置包括以下详细信息，您可以在创建管道后对其进行编辑：

- 您的分配设置的描述。
- 您分配映像的区域的区域设置。区域 1 默认为您创建管道的区域。您可以使用添加区域按钮添加要分配的区域，也可以删除除区域 1 之外的所有区域。

  区域设置包括：

  - 目标区域。
  - 该服务默认为 “ECR”，并且不可编辑。
  - 存储库名称-目标存储库的名称（不包括 Amazon ECR 位置）。例如，带有位置的存储库名称将如下所示：

    ```
    <account-id>.dkr.ecr.<region>.amazonaws.com/<repository-name>
    ```

> **Note**
>
> 如果更改存储库名称，则只有名称更改后创建的映像才会添加到新名称下。您管道之前创建的所有映像都将保留在其原始存储库中。

要从管道详细信息页面编辑您的分配设置，请按照以下步骤操作：

1. 选择分配设置选项卡。
2. 从分配详细信息面板的右上角选择编辑。
3. 当您准备好保存对分配设置所做的更新时，请选择保存更改。

编辑管道的构建计划

编辑管道页面包含以下详细信息，您可以在创建管道后对其进行编辑：

- 对您管道的描述。
- 增强型元数据收集。默认情况下，此选项处于打开状态。要将其关闭，请清除启用增强型元数据收集复选框。
- 您管道的构建计划。您可以在此节中更改您的计划选项和所有设置。

要从管道详细信息页面编辑您的管道，请按照以下步骤操作：

1. 在管道详细信息页面的右上角，选择操作，然后选择编辑管道。
2. 准备好保存更新后，选择保存更改。

> **Note**
>
> 有关使用 cron 表达式计划构建的更多信息，请参阅 在 Image Builder 中使用 cron 表达式。

从中更新容器镜像管道 AWS CLI

可以使用 JSON 文件作为 AWS CLI中 update-image-pipeline 命令的输入来更新容器映像管道。要配置 JSON 文件，您必须有 Amazon 资源名称 (ARNs) 才能引用以下现有资源：

- 要更新的映像管道
- 容器配方
- 基础设施配置
- 分配设置（如果包含在当前管道中）

> **Note**
>
> 如果包含分配设置资源，则在运行命令的区域（区域 1）的分配设置中指定为目标存储库的 ECR 存储库优先级将高于容器配方中指定的目标存储库。

按照以下步骤使用 AWS CLI中的 update-image-pipeline 命令更新容器映像管道：

> **Note**
>
> UpdateImagePipeline 不支持对管道进行选择性更新。您必须在更新请求中指定所有必需的属性，而不仅仅是已更改的属性。

1. 创建 CLI 输入 JSON 文件

   使用您常用的文件编辑工具创建 JSON 文件，其中包含以下密钥以及对您的环境有效的值。此示例使用名为 `create-component.json` 的文件：

```
{
 "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
pipeline/my-example-pipeline",
 "containerRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:container-
recipe/my-example-recipe/2020.12.08",
 "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"
}
```

> **Note**
>
> - JSON 文件路径开头必须包含 `file://` 符号。
> - JSON 文件的路径应遵循运行命令的基本操作系统的相应约定。例如，Windows 使用反斜杠 (\) 引用目录路径，而 Linux 和 macOS 使用正斜杠 (/)。

2. 使用创建的文件作为输入，运行以下命令。

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

在 Image Builder 配置映像管道工作流

借助映像工作流，您可以根据需要自定义管道运行的工作流程，以构建和测试映像。您定义的工作流在 Image Builder 工作流框架的上下文中运行。有关构成工作流框架的各个阶段的更多信息，请参阅管理 Image Builder 映像的构建和测试工作流。

构建工作流

构建工作流在工作流框架的 `Build` 阶段运行。只能为管道指定一个构建工作流。或者可以完全跳过构建来配置仅测试管道。

测试工作流

测试工作流在工作流框架的 `Test` 阶段运行。最多可以为管道指定十个测试工作流。如果只想要构建管道，也可以完全跳过测试。

定义测试工作流的测试组

测试工作流在测试组中进行定义。最多可以为管道运行十个测试工作流。您可以决定是按特定顺序运行测试工作流，还是同时运行尽可能多的测试工作流。其运行方式取决于您如何定义测试组。以下场景演示了几种定义测试工作流的方法。

> **Note**
>
> 如果您使用控制台创建工作流，我们建议您在定义测试组之前，花时间计划运行测试工作流的运行方式。在控制台中，您可以添加或删除测试工作流和组，但不能对其进行重新排序。

场景 1：一次运行一个测试工作流

要逐一运行所有测试工作流，您最多可以配置十个测试组，每个测试组中包含一个测试工作流。测试组按您将其添加到管道的顺序逐一运行。这是确保测试工作流按特定顺序逐一运行的一种方法。

场景 2：同时运行多个测试工作流

如果顺序无关紧要，并且您希望同时运行尽可能多的测试工作流，则可以配置单个测试组，并在其中放入最大数量的测试工作流。Image Builder 可同时启动多达五个测试工作流，并在其他工作流完成后启动其他测试工作流。如果您的目标是尽可能快地运行测试工作流，那么这是一种实现目标的方法。

场景 3：混合搭配

如果是混合场景，其中有些测试工作流可以同时运行，而另一些则应该一次运行一个，那么您可以配置测试组来实现这个目标。如何配置测试组的唯一限制是可以为管道运行的测试工作流的最大数量

通过控制台在 Image Builder 管道中设置工作流参数

对于构建工作流和测试工作流，工作流参数的作用方式相同。创建或更新管道时，您可以选择希望包含的构建和测试工作流。如果在工作流文档中为所选工作流定义了参数，则 Image Builder 会在参数面板中显示这些参数。对于未定义参数的工作流，该面板处于隐藏状态。

每个参数都显示工作流文档定义的以下属性：

- 名称（不可编辑）– 参数的名称。
- 类型（不可编辑）- 参数值的数据类型。
- 值 – 参数的值。您可以编辑参数值以设置管道的参数值。

指定 Image Builder 用于运行工作流操作的 IAM 服务角色

要运行映像工作流，Image Builder 需要获得执行工作流操作的权限。您可以指定 AWSServiceRoleForImageBuilder 服务相关角色，也可以为服务访问指定自己的自定义角色，如下所示。

- 控制台 – 在管道向导步骤 3 定义映像创建过程中，从服务访问面板的 IAM 角色列表中选择服务相关角色或您自己的自定义角色。
- I@@ mage Builder API — 在CreateImage操作请求中，将服务相关角色或您自己的自定义角色指定为`executionRole`参数值。

要详细了解如何创建服务角色，请参阅AWS Identity and Access Management 用户指南中的创建角色以向 AWS 服务委派权限。

在 Image Builder 管道中使用 EventBridge 规则

来自各种合作伙伴服务的活动几乎实时地 AWS 流式传输到Amazon EventBridge 活动总线。您还可以生成自定义事件，并将事件从您自己的应用程序发送到 EventBridge。事件总线使用规则来确定将事件数据路由到何处。

Image Builder 管道可用作 EventBridge 规则目标，这意味着您可以根据为响应总线上的事件而创建的规则或按计划运行 Image Builder 管道。

有关 Image Builder 发送到的系统生成事件的摘要 EventBridge，请参阅Image Builder 发送的事件消息。

> **Note**
>
> 事件总线是特定于某个区域的。规则和目标必须位于同一区域。

内容

EventBridge 条款

本节包含术语摘要，可帮助您了解如何 EventBridge 与 Image Builder 管道集成。

事件

描述环境中可能影响一个或多个应用程序资源的变化。环境可以是 AWS 环境、SaaS 合作伙伴服务或应用程序，也可以是您的一个应用程序或服务。您还可以在时间线上设置计划的事件。

事件总线

接收来自应用程序和服务的事件数据的管道。

来源

将事件发送到事件总线的服务或应用程序。

目标

一种资源或端点，在匹配规则时 EventBridge 调用，将数据从事件传送到目标。

规则

规则 规则匹配传入事件并将其路由到目标进行处理。一条规则可以将一个事件发送到多个目标，然后这些目标将可并行运行。规则要么基于事件模式，要么基于计划。

模式

事件模式定义事件结构和规则匹配的字段，以启动目标操作。

计划

计划规则在计划中执行操作，例如运行 Image Builder 管道在每个季度刷新映像。有两种类型的计划表达式：

- Cron 表达式 — 使用可以概述简单条件的 cron 语法匹配特定的调度条件；例如，每周在特定日期运行。您还可以制定更复杂的条件，例如每季度在每月的第五天凌晨 2 点到凌晨 4 点之间运行。
- rate 表达式-指定调用目标时的固定间隔，例如每 12 小时一次。

查看 Image Builder 管道的 EventBridge 规则

Image Builder Image 管道详细信息页面中的EventBridge 规则选项卡显示 EventBridge 您的账户有权访问的事件总线，以及适用于当前管道的所选事件总线规则。此选项卡还直接链接到用于创建新资源的 EventBridge 控制台。

链接到 EventBridge 控制台的操作

- 创建事件总线
- 创建规则

要了解更多信息 EventBridge，请参阅 Amazon EventBridge 用户指南中的以下主题。

- 什么是亚马逊 EventBridge
- 亚马逊 EventBridge 活动巴士
- 亚马逊 EventBridge 活动
- 亚马逊 EventBridge 规则

使用 EventBridge 规则安排管道构建

在本示例中，我们使用 rate 表达式为默认事件总线创建了新的计划规则。此示例中的规则每 90 天在事件总线上生成一个事件。该事件启动管道构建以刷新图像。

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 要查看在您的账户下创建的映像管道列表，请从导航窗格中选择映像管道。

   > **Note**
   >
   > 映像管道列表包括管道创建的输出映像类型的指示器，即 AMI 或 Docker。

3. 要查看详细信息或编辑管道，请选择管道名称链接。这将打开管道的详细视图。

   > **Note**
   >
   > 您也可以选中管道名称旁边的框，然后选择查看详细信息。

4. 打开 “EventBridge 规则” 选项卡。
5. 保留在事件总线面板中预先选择的默认事件总线。
6. 选择创建规则。这将带您进入亚马逊 EventBridge 控制台中的创建规则页面。
7. 为规则输入名称和描述。规则名称在所选区域的事件总线中必须是唯一的。
8. 在定义模式面板中，选择计划选项。这将展开面板，每个选项都选中了 Fixed rate every。
9. 在第一个框中输入 90，然后从下拉列表中选择Days。

10. 在选择目标面板中执行以下操作：

    a. 从目标 下拉列表中选择 `EC2 Image Builder`

    b. 要将规则应用于 Image Builder 管道，请从映像管道下拉列表中选择目标管道。

    c. EventBridge 需要权限才能启动所选管道的构建。在此示例中，保留为此特定资源创建新角色的默认选项。

    d. 选择 Add target。

11. 选择 Create（创建）。

> **Note**
>
> 要详细了解本示例中未涉及的费率表达式规则设置，请参阅 Amazon EventBridge 用户指南中的费率表达式。

将产品和服务集成到 Image Builder 中

EC2 Image Builder 与其他 AWS 服务 应用程序集成，可帮助您创建强大、安全的自定义计算机映像。AWS Marketplace

产品

Image Builder 配方可以合并来自 AWS Marketplace Image Builder 托管组件的图像产品，以提供专门的构建和测试功能，如下所示。

- AWS Marketplace 图片产品-使用来自的图片产品 AWS Marketplace 作为配方中的基础图片，以满足组织标准，例如 CIS Hardening。 从 Image Builder 控制台创建配方时，您可以从现有订阅中进行选择，也可以从 AWS Marketplace中搜索特定产品。当您通过 Image Builder API、CLI 或 SDK 创建配方时，您可以指定映像产品 Amazon 资源名称 (ARN) 作为基础映像。
- Image Builder 组件 – 您在配方中指定的组件可以执行构建和测试操作，例如，安装软件或执行合规性验证。您从 AWS Marketplace 订阅的某些映像产品可能包含可在配方中使用的配套组件。CIS Hardened 映像包含一个匹配的 AWSTOE 组件，您可以在配方中使用该组件来强制执行 CIS 基准第 1 级指导方针。

> **Note**
>
> 有关合规相关产品的更多信息，请参阅 适用于您的 Image Builder 映像的合规产品。

服务

Image Builder 与以下内容 AWS 服务 集成，可提供详细的事件指标、日志记录和监控。此信息可帮助您跟踪活动、解决映像构建问题以及根据事件通知创建自动化。

- AWS Organizations— AWS Organizations 允许您对组织中的帐户应用服务控制策略 (SCP)。您可以创建、管理、启用和禁用各个策略。与所有其他 AWS 工件和服务类似，Image Builder 也遵循中定义的策略 AWS Organizations。 AWS SCPs 为常见场景提供了模板，例如对成员账户实施限制，使其只能在获得批准的情况下启动实例 AMIs。
- AWS CloudTrail— 监控发送到的 Image Builder 事件 CloudTrail。有关与 Image Builder CloudTrail 集成的更多信息，请参阅使用记录 Image Builder API 调用 CloudTrail。

  要了解更多信息 CloudTrail，包括如何将其开启和查找日志文件，请参阅AWS CloudTrail 用户指南。

- Amazon CloudWatch 日志 — 使用监控、存储和访问您的 Image Builder 日志文件 CloudWatch。您还可以选择将日志保存到 S3 存储桶。要了解有关与 Image Builder CloudWatch 集成的更多信息，请参阅使用 Amazon 日志监控 Image Builder CloudWatch 日志。

  有关 CloudWatch 日志的更多信息，请参阅什么是 Amazon CloudWatch 日志？ 在 Amazon CloudWatch 日志用户指南中。

- Amazon Elastic Container Registry (Amazon ECR) - Amazon ECR 是一项安全、可靠且可扩展的 AWS 托管容器映像注册表服务。使用 Image Builder 创建的容器映像会存储在源区域（构建运行的区域）和分配容器映像的任何区域的 Amazon ECR 中。有关 Amazon ECR 的更多信息，请参阅 Amazon Elastic Container Registry 用户指南。

- Amazon EventBridge — Connect 连接到来自您账户中 Image Builder 活动的实时事件数据流。有关的更多信息 EventBridge，请参阅 Amazon 是什么 EventBridge？ 在《亚马逊 EventBridge 用户指南》中。

- Amazon Inspector – 通过自动扫描 Image Builder 在您创建新映像时启动的 EC2 测试实例，发现软件和网络设置中的漏洞。Image Builder 会保存输出映像资源的调查发现，以便您可以在测试实例终止后进行调查和修复。有关扫描和定价的更多信息，请参阅《Amazon Inspector 用户指南》中的 What is Amazon Inspector?

  如果您配置了增强扫描，Amazon Inspector 也可以扫描您的 ECR 存储库。有关更多信息，请参阅 Amazon Inspector 用户指南中的扫描 Amazon ECR 容器映像。

  > **Note**
  >
  > Amazon Inspector 是一项付费功能。

- AWS License Manager – 您可以在分配过程中将 License Manager 自管理许可证附加到输出 AMI。您为目标区域指定的许可证必须已在该区域中存在。有关更多信息，请参阅 Self-managed licenses in License Manager。

- AWS Marketplace — 查看您当前的 AWS Marketplace 产品订阅列表，并直接从 Image Builder 中搜索映像产品。您也可以使用已订阅的映像产品作为 Image Builder 配方的基础映像。有关管理 AWS Marketplace 订阅的更多信息，请参阅《买AWS Marketplace 家指南》中的 “购买商品”。

- AWS Resource Access Manager (AWS RAM) — 使用 AWS RAM，您可以与任何人共享资源 AWS 账户 或通过共享资源 AWS Organizations。如果您有多个资源 AWS 账户，则可以集中创建资源 AWS RAM 并使用这些资源与其他账户共享。EC2 Image Builder 允许共享以下资源：组件、映像和映像配方。有关的更多信息 AWS RAM，请参阅《AWS Resource Access Manager 用户指南》。有关共享 Image Builder 资源的信息，请参阅 与共享 Image Builder 资源 AWS RAM。

- Amazon Simple Notification Service (Amazon SNS) — 如果已配置，请将有关您的映像状态的详细消息发布到您订阅的 SNS 主题中。有关 Amazon SNS 的更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的什么是 Amazon SNS？

产品和服务集成主题

- 亚马逊 EventBridge 集成到 Image Builder 中
- Image Builder 中的 Amazon Inspector 集成
- AWS Marketplace 在 Image Builder 中集成
- Image Builder 中的 Amazon SNS 集成
- 适用于您的 Image Builder 映像的合规产品

亚马逊 EventBridge 集成到 Image Builder 中

Amazon EventBridge 是一项无服务器事件总线服务，您可以使用它来连接您的 Image Builder 应用程序与其他 AWS 服务应用程序的相关数据。在中 EventBridge，规则匹配传入的事件并将其发送到目标进行处理。一条规则可以将一个事件发送到多个目标，然后这些事件并行运行。

借 EventBridge助，您可以自动执行 AWS 服务 并自动响应系统事件，例如应用程序可用性问题或资源更改。来自 AWS 服务 的事件以近乎实时 EventBridge 的方式传送到。您可以设置对传入事件做出响应的规则以启动操作。例如，当 EC2 实例的状态从待处理变为正在运行时，发送事件到 Lambda 函数。这些被称为模式。要基于事件模式创建规则，请参阅 EventBridge 《亚马逊 EventBridge 用户指南》中的创建对事件做出反应的亚马逊规则。

可自动触发的操作包括：

- 调用一个 AWS Lambda 函数
- 调用 Amazon EC2 Run Command
- 将事件中继到 Amazon Kinesis Data Streams
- 激活 AWS Step Functions 状态机
- 通知 Amazon SNS 主题或 Amazon SQS 队列

您还可以为默认事件总线设置调度规则，使其定期执行操作，例如运行 Image Builder 管道以每季度刷新一次映像。有两种类型的计划表达式：

- cron 表达式 — 以下 cron 表达式示例将任务安排在每天中午 (UTC+0) 运行：

```
cron(0 12 * * ? *)
```

有关将 cron 表达式与一起使用的更多信息 EventBridge，请参阅 A mazon EventBridge 用户指南中的 Cron 表达式。

- rate 表达式 — 以下 rate 表达式示例将任务安排为每 12 小时运行一次：

```
rate(12 hour)
```

有关在中使用费率表达式的更多信息 EventBridge，请参阅 Amazon EventBridge 用户指南中的费率表达式。

有关 EventBridge 规则如何与 Image Builder 图像管道集成的更多信息，请参阅在 Image Builder 管道中使用 EventBridge 规则。

Image Builder 发送的事件消息

当 Image Builder 资源的状态发生重大变化 EventBridge 时，Image Builder 会向其发送事件消息。例如，当映像的状态发生变化时。以下示例显示了 Image Builder 可能发送的典型 JSON 事件消息。

主题

- EC2 Image Builder Image State Change
- EC2 Image Builder CVE Detected
- EC2 Image Builder Workflow Step Waiting
- EC2 Image Builder 镜像管道自动禁用

EC2 Image Builder Image State Change

在映像创建过程中，当映像资源的状态发生变化时，Image Builder 会发送此事件。例如，当映像状态从一种状态变为另一种状态时，如下所示：

- 从 `building` 到 `testing`
- 从 `testing` 到 `distribution`
- 从 `testing` 到 `failed`
- 从 `integrating` 到 `available`

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Image State Change",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2024-01-18T17:50:56Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image/
cmkencryptedworkflowtest-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/1.0.0/1"],
    "detail": {
        "previous-state": {
            "status": "TESTING"
        },
        "state": {
            "status": "AVAILABLE"
        }
    }
}
```

EC2 Image Builder CVE Detected

如果您为映像启用了 CVE 检测，则每当完成映像扫描时，Image Builder 都会发送一条包含扫描结果的消息。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder CVE Detected",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2023-03-01T16:59:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:imagebuilder:us-east-1:111122223333:image/test-image/1.0.0/1",
        "arn:aws:imagebuilder:us-east-1:111122223333:image-pipeline/test-pipeline"
    ],
    "detail": {
        "resource-id": "i-1234567890abcdef0",
        "finding-severity-counts": {
            "all": 0,
            "critical": 0,
            "high": 0,
```

```
            "medium": 0
        }
    }
}
```

EC2 Image Builder Workflow Step Waiting

当 WaitForAction 工作流步骤暂停以等待异步操作完成时，Image Builder 会发送一条消息。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Workflow Step Waiting",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2024-01-18T16:54:44Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image/
workflowstepwaitforactionwithvalidsnstopictest-a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222/1.0.0/1", "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
build-workflow-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333/1.0.0/1"],
    "detail": {
        "workflow-execution-id": "wf-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "workflow-step-execution-id": "step-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "workflow-step-name": "TestAutoSNSStop"
    }
}
```

EC2 Image Builder 镜像管道自动禁用

如果您已autoDisablePolicy为管道配置了，则 Image Builder 会禁用管道，并在连续调度管道执行失败次数超过策略允许的最大数量 EventBridge 时向其发送事件消息。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Image Pipeline Automatically Disabled",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2025-09-18T16:54:44Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image-pipeline/disabled-
image-pipeline-name"],
```

```
    "detail": {
        "consecutive-failures": "5"
    }
}
```

Image Builder 中的 Amazon Inspector 集成

当您使用 Amazon Inspector 激活安全扫描时，它会持续扫描您账户中的计算机映像和正在运行的实例，以查找操作系统和编程语言的漏洞。如果激活，则安全扫描将自动进行，Image Builder 可以在您创建新映像时保存测试实例中的调查发现快照。Amazon Inspector 是一项付费服务。

当 Amazon Inspector 发现您的软件或网络设置中存在的漏洞时，它会采取以下措施：

- 通知您有调查发现。
- 评估结果的严重性。严重性评级对漏洞进行分类，以帮助您确定调查发现的优先级，并包括以下值：
  - 未分类
  - 信息性
  - 低
  - 中
  - 高
  - 重大
- 提供有关调查发现的信息，并提供指向其他资源的链接以获取更多详细信息。
- 提供补救指导，帮助您解决导致调查发现的问题。

配置安全扫描

如果您已为自己的账户激活了 Amazon Inspector，Amazon Inspector 会自动扫描 Image Builder 启动的 EC2 实例，以构建和测试新映像。在构建和测试过程中，这些实例的生命周期很短，它们的调查发现通常会在这些实例关闭后立即过期。为了帮助您调查和修复新映像的调查发现，Image Builder 可以选择将 Amazon Inspector 在构建过程中在测试实例上识别的任何调查发现保存为快照。

要为您的管道配置安全扫描，请参阅 在中为 Image Builder 图像配置安全扫描 AWS 管理控制台。

查看安全调查发现

在 Image Builder 控制台中，您可以集中查看所有 Image Builder 资源的安全调查发现。您可以在安全概述部分的安全调查发现页面上查看所有调查发现，也可以按漏洞、映像管道或映像对调查发现进行分

组。控制台默认显示所有安全调查发现。所有安全调查发现选项的摘要面板显示每个严重性级别的调查发现数量。有关更多信息，请参阅 在中管理 Image Builder 图像的安全发现 AWS 管理控制台。

要了解有关 Amazon Inspector 漏洞调查发现的更多信息，请参阅 Amazon Inspector 用户指南中的了解 Amazon Inspector 中的调查发现。

AWS Marketplace 在 Image Builder 中集成

AWS Marketplace 是一个精心策划的数字目录，您可以在其中查找和订阅第三方软件、数据和服务，这些软件、数据和服务可帮助您构建满足业务需求的解决方案。 AWS Marketplace 汇集了经过身份验证的买家和注册卖家，以及来自安全、网络、存储、机器学习等热门类别的软件清单。

AWS Marketplace 卖方可以是独立软件供应商 (ISV)、经销商，也可以是可以提供适用于 AWS 产品和服务的个人。当卖家提交产品时 AWS Marketplace，他们会定义产品的价格以及使用条款和条件。买家同意定价，以及为报价设定的条款和条件。要了解更多信息 AWS Marketplace，请参阅什么是 AWS Marketplace？

AWS Marketplace 集成功能

Image Builder AWS Marketplace 与集成，可直接从 Image Builder 控制台提供以下功能：

- 搜索中可用的图片商品 AWS Marketplace。
- 搜索提供组件的 AWS Marketplace 图片商品。
- 查看您当前 AWS Marketplace 的产品订阅列表。
- 使用您已订阅的 AWS Marketplace 图片产品作为 Image Builder 配方的基础图片。
- 使用您在 Image Builder 配方中订阅的 AWS Marketplace 组件。

Image Builder 与 AWS Marketplace 集成，可显示您订阅的图像产品和组件。您也可以在不退出 AWS Marketplace Image Builder 控制台的情况下从 “发现产品” 页面搜索图片产品和组件。

Image Builder 创建的输出 AMI 包括来自 AWS Marketplace 图像产品和组件的产品代码。您最多可以为最终的自定义图片设置四个产品代码。

AWS Marketplace Image Builder 中的订阅

Image Builder 控制台 AWS Marketplace 部分的订阅页面会显示您当前订阅 AWS Marketplace 的产品列表。每个订阅的产品都显示以下详细信息：

- 产品名称。这链接到中的商品详情页面 AWS Marketplace。您订阅的产品的产品详情页面将在浏览器的新选项卡中打开。
- 出版商。它链接到中的出版商详情页面 AWS Marketplace。此时将在浏览器的新选项卡中打开发布者详情页面。
- 您订阅的版本。
- 如果您订阅的产品中包含任何关联组件，Image Builder 会显示指向组件详细信息的链接。

在页面顶部，您可以按名称搜索特定产品，也可以使用分页控件按页翻阅结果。要在新食谱中使用已订阅的图片产品，请选择已订阅的产品并选择创建新食谱。默认情况下，Image Builder 会预先选择列表中的第一个产品。

> **Note**
>
> 如果您要寻找刚才订阅的产品，但未在列表中看到该产品，请使用选项卡顶部的刷新按钮刷新搜索结果。新订阅可能需要几分钟才会在列表中显示。

通过 AWS Marketplace Image Builder 控制台发现图片产品

本节重点介绍在 AWS Marketplace 食谱中用作基础图片的图片产品。对于包含关联软件组件的产品，您可以在控制台以及 API、SDK 和 CLI 中筛选产品所有者。有关更多信息，请参阅 列出 Image Builder 组件。有关查找、订阅和使用 AWS Marketplace 组件的更多信息，请参阅使用 AWS Marketplace 组件自定义您的图像。

探索产品

要从 I AWS Marketplace mage Builder 控制台中查找图片产品，请按照以下步骤操作：

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择AWS Marketplace部分中的发现产品。
3. 您可以在发现商品页面的图片商品选项卡中搜索图片商品。

   Image Builder 会对产品 AWS Marketplace 进行预过滤，将重点放在您可以在 Image Builder 配方中使用的机器图像上。有关与 Image Builder AWS Marketplace 集成的更多信息，请选择与您想要查看的内容相匹配的选项卡。

   该选项卡包含两个面板。左侧的优化结果面板可以帮助您筛选结果以查找要订阅的产品。右侧的搜索产品面板显示了符合筛选条件的产品，还提供了按产品名称进行搜索的选项。

优化结果

以下列表仅显示了可以应用于产品搜索的几个筛选条件：

- 选择一个或多个产品类别，例如基础设施软件或机器学习。
- 为您的映像产品选择操作系统，或者为特定操作系统平台选择所有产品，例如 AllLinux/Unix。
- 选择一个或多个发布者来显示其可用产品。选择显示全部链接，以显示产品符合您所应用筛选条件的所有发布者。

> **Note**
>
> 发布者名称不按字母顺序显示。如果要查找特定的发布者，比如 `Center for Internet Security`，可以在所有发布者对话框顶部的搜索框中输入其名称的一部分。应该把名称拼写出来，比如 `CIS` 可能不会产生您想要的结果。
> 您也可以逐页浏览发布者名称。

筛选条件选项是动态选项。您所做的每个选择都会影响您对所有其他类别的选择。有成千上万种产品可供选择 AWS Marketplace，因此你可以筛选的越多，找到你想要的东西的可能性就越大。

搜索产品

要按名称查找特定产品，您可以在此面板顶部的搜索栏中输入名称的一部分。每个产品结果都包含以下详细信息：

- 产品名称和徽标。这两者都链接到 AWS Marketplace中的产品详情页面。详细信息页面将在浏览器的新选项卡中打开。如果您想在 Image Builder 配方中使用映像产品，则可以从此处订阅该映像产品。有关更多信息，请参阅 AWS Marketplace 买家指南中的购买产品。

  如果您在中订阅了图片产品 AWS Marketplace，请切换回浏览器中的 Image Builder 选项卡，然后刷新已订阅的图片产品列表以进行查看。

  > **Note**
  >
  > 可能需要几分钟时间，新订阅才能生效。

- 发布者名称。它链接到中的出版商详情页面 AWS Marketplace。此时将在浏览器的新选项卡中打开发布者详情页面。

- 产品版本。
- 商品星级，以及直接指向 AWS Marketplace中产品详情页面检查部分的链接。详细信息页面将在浏览器的新选项卡中打开。
- 产品描述的前几行。

在搜索栏的正下方，可以看到您的搜索产生了多少结果，以及当前显示了那些结果的哪些子集。您可以使用面板右侧的其他控件来调整设置每次显示的产品数量，并调整应用于结果的排序顺序。您还可以使用分页控件按页翻阅结果。

在 AWS Marketplace Image Builder 食谱中使用图片产品

打开创建食谱页面，选择要用作基础 AWS Marketplace 图片的图片产品，如下所示。

1. 打开 EC2 Image Builder 控制台，网址为https://console.aws.amazon.com/imagebuilder/。
2. 在导航窗格中，选择 AWS Marketplace 部分中的映像配方。此时会显示您创建的映像配方列表。
3. 选择创建映像配方。此时将打开创建配方页面。
4. 如同往常一样，在配方详细信息部分中，输入您的配方名称和版本。
5. 在基础映像部分中，选择AWS Marketplace 映像选项。这会在 “订阅” 选项卡中显示您已订阅的 AWS Marketplace 图片商品的列表。您可以从列表中选择基础映像。

   您也可以 AWS Marketplace 直接从该AWS Marketplace选项卡中搜索其他可用的图片商品。选择添加产品，或者直接打开 AWS Marketplace 选项卡。有关如何设置筛选条件和在中进行搜索的更多信息 AWS Marketplace，请参阅通过 AWS Marketplace Image Builder 控制台发现图片产品。
6. 照常输入其余细节。如果您的任何产品订阅包含构建组件，则可以从构建组件列表中选择它们。AWS Marketplace从组件所有者类型列表中选择以查看它们，或者选择 Third party managed CIS 组件。
7. 选择创建配方。

   您的最终图片最多可以包含来自 AWS Marketplace 图片产品和组件的四个产品代码。如果您选择的基本图片和组件包含四个以上的产品代码，则当您尝试创建配方时，Image Builder 会返回错误。

Image Builder 中的 Amazon SNS 集成

Amazon Simple Notification Service (Amazon SNS) 是一项托管服务，提供从发布者向订阅者（也称为创建者和使用者）的异步消息传输。

您可以在基础设施配置中指定 SNS 主题。当您创建映像或运行管道时，Image Builder 可以向该主题发布有关您的映像状态的详细消息。当映像状态达到以下状态之一时，Image Builder 会发布一条消息：

- `AVAILABLE`
- `FAILED`

有关来自 Image Builder 的 SNS 消息示例，请参阅 SNS 消息格式。如果您想要创建新的 SNS 主题，请参阅 Amazon Simple Notification Service 开发人员指南中的 Amazon SNS 入门。

已加密的 SNS 主题

如果您的 SNS 主题已加密，则必须在 AWS KMS key 策略中授予 Image Builder 服务角色执行以下操作的权限：

- `kms:Decrypt`
- `kms:GenerateDataKey`

> **Note**
>
> 如果 SNS 主题已加密，则加密此主题的密钥必须位于 Image Builder 服务运行的账户中。Image Builder 无法向使用其他账户密钥进行加密的 SNS 主题发送通知。

KMS 密钥策略添加示例

以下示例显示了您添加到 KMS 密钥策略的额外部分。将 Amazon 资源名称 (ARN) 用于您首次创建 Image Builder 映像时 Image Builder 在您的账户下创建的 IAM 服务相关角色。要了解有关 Image Builder 的服务相关角色的更多信息，请参阅 使用 Image Builder 的 IAM 服务相关角色。

```
{
  "Statement": [{
```

```
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::123456789012:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

您可以使用以下其中一种方法来获取 ARN。

AWS 管理控制台

要从中获取 Image Builder 在您的账户下创建的服务相关角色的 ARN AWS 管理控制台，请执行以下步骤：

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在左侧导航窗格中，选择 角色。
3. 搜索 ImageBuilder 并从结果中选择以下角色名称：AWSServiceRoleForImageBuilder。此时将显示角色详情页面。
4. 要将 ARN 复制到剪贴板，请选择 ARN 名称旁边的图标。

AWS CLI

要从中获取 Image Builder 在您的账户下创建的服务相关角色的 ARN AWS CLI，请按如下方式使用 IAM get- role 命令。

```
aws iam get-role --role-name AWSServiceRoleForImageBuilder
```

部分示例输出：

```
{
    "Role": {
        "Path": "/aws-service-role/imagebuilder.amazonaws.com/",
        "RoleName": "AWSServiceRoleForImageBuilder",
        ...
```

```
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder",
        ...
}
```

SNS 消息格式

在 Image Builder 向您的 Amazon SNS 主题发布消息后，订阅该主题的其他服务可以对消息格式进行筛选并确定其是否符合进一步操作的标准。例如，成功消息可能会启动更新 AWS Systems Manager 参数存储或为输出 AMI 启动外部合规性测试工作流程的任务。

以下示例显示了 Image Builder 在管道构建运行至完成时发布的典型消息的 JSON 负载，并创建了一个 Linux 映像。

```
{
  "versionlessArn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image",
  "semver": 1237940039285380274899124227,
  "arn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image/1.0.0/3",
  "name": "example-linux-image",
  "version": "1.0.0",
  "type": "AMI",
  "buildVersion": 3,
  "state": {
    "status": "AVAILABLE"
  },
  "platform": "Linux",
  "imageRecipe": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:image-recipe/example-linux-
image/1.0.0",
    "name": "amjule-barebones-linux",
    "version": "1.0.0",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-west-1:123456789012:component/update-
linux/1.0.2/1"
      }
    ],
    "platform": "Linux",
    "parentImage": "arn:aws:imagebuilder:us-west-1:987654321098:image/amazon-linux-2-
x86/2022.6.14/1",
```

```
    "blockDeviceMappings": [
      {
        "deviceName": "/dev/xvda",
        "ebs": {
          "encrypted": false,
          "deleteOnTermination": true,
          "volumeSize": 8,
          "volumeType": "gp2"
        }
      }
    ],
    "dateCreated": "Feb 24, 2021 12:31:54 AM",
    "tags": {
      "internalId": "1a234567-8901-2345-bcd6-ef7890123456",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:image-recipe/example-
linux-image/1.0.0"
    },
    "workingDirectory": "/tmp",
    "accountId": "462045008730"
  },
  "sourcePipelineArn": "arn:aws:imagebuilder:us-west-1:123456789012:image-pipeline/
example-linux-pipeline",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:infrastructure-configuration/
example-linux-infra-config-uswest1",
    "name": "example-linux-infra-config-uswest1",
    "instanceProfileName": "example-linux-ib-baseline-admin",
    "tags": {
      "internalId": "234abc56-d789-0123-a4e5-6b789d012c34",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:infrastructure-
configuration/example-linux-infra-config-uswest1"
    },
    "logging": {
      "s3Logs": {
        "s3BucketName": "amzn-s3-demo-bucket"
      }
    },
    "keyPair": "example-linux-key-pair-uswest1",
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-1:123456789012:example-linux-ibnotices-
uswest1",
    "dateCreated": "Feb 24, 2021 12:31:55 AM",
    "accountId": "123456789012"
  },
```

```
  "imageTestsConfigurationDocument": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
  },
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:distribution-configuration/
example-linux-distribution",
    "name": "example-linux-distribution",
    "dateCreated": "Feb 24, 2021 12:31:56 AM",
    "distributions": [
      {
        "region": "us-west-1",
        "amiDistributionConfiguration": {}
      }
    ],
    "tags": {
      "internalId": "345abc67-8910-12d3-4ef5-67a8b90c12de",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:distribution-
configuration/example-linux-distribution"
    },
    "accountId": "123456789012"
  },
  "dateCreated": "Jul 28, 2022 1:13:45 AM",
  "outputResources": {
    "amis": [
      {
        "region": "us-west-1",
        "image": "ami-01a23bc4def5a6789",
        "name": "example-linux-image 2022-07-28T01-14-17.416Z",
        "accountId": "123456789012"
      }
    ]
  },
  "buildExecutionId": "ab0cd12e-34fa-5678-b901-2c3456d789e0",
  "testExecutionId": "6a7b8901-cdef-234a-56b7-8cd89ef01234",
  "distributionJobId": "1f234567-8abc-9d0e-1234-fa56b7c890de",
  "integrationJobId": "432109b8-afe7-6dc5-4321-0ba98f7654e3",
  "accountId": "123456789012",
  "osVersion": "Amazon Linux 2",
  "enhancedImageMetadataEnabled": true,
  "buildType": "USER_INITIATED",
  "tags": {
    "internalId": "901e234f-a567-89bc-0123-d4e567f89a01",
```

```
    "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image/1.0.0/3"
  }
}
```

以下示例显示了 Image Builder 为 Linux 映像的管道构建失败而发布的典型消息的 JSON 负载。

```
{
  "versionlessArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image",
  "semver": 12379400392853802748991242 31,
  "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-image/1.0.0/7",
  "name": "My Example Image",
  "version": "1.0.0",
  "type": "AMI",
  "buildVersion": 7,
  "state": {
    "status": "FAILED",
    "reason": "Image Failure reason."
  },
  "platform": "Linux",
  "imageRecipe": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
image/1.0.0",
    "name": "My Example Image",
    "version": "1.0.0",
    "description": "Testing Image recipe",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-
example-image-component/1.0.0/1"
      }
    ],
    "platform": "Linux",
    "parentImage": "ami-0cd12345db678d90f",
    "dateCreated": "Jun 21, 2022 11:36:14 PM",
    "tags": {
      "internalId": "1a234567-8901-2345-bcd6-ef7890123456",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-
example-image/1.0.0"
    },
    "accountId": "123456789012"
  },
```

```
  "sourcePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-image-pipeline",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-infra-config",
    "name": "SNS topic Infra config",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
      "t2.micro"
    ],
    "instanceProfileName": "EC2InstanceProfileForImageBuilder",
    "tags": {
      "internalId": "234abc56-d789-0123-a4e5-6b789d012c34",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-infra-config"
    },
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:example-pipeline-notification-topic",
    "dateCreated": "Jul 5, 2022 7:31:53 PM",
    "accountId": "123456789012"
  },
  "imageTestsConfigurationDocument": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
  },
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-distribution-config",
    "name": "New distribution config",
    "dateCreated": "Dec 3, 2021 9:24:22 PM",
    "distributions": [
      {
        "region": "us-west-2",
        "amiDistributionConfiguration": {},
        "fastLaunchConfigurations": [
          {
            "enabled": true,
            "snapshotConfiguration": {
              "targetResourceCount": 2
            },
            "maxParallelLaunches": 2,
            "launchTemplate": {
              "launchTemplateId": "lt-01234567890"
```

```
            },
            "accountId": "123456789012"
          }
        ]
      }
    ],
    "tags": {
      "internalId": "1fecd23a-4f56-7f89-01e2-345678abbe90",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-
configuration/my-example-distribution-config"
    },
    "accountId": "123456789012"
  },
  "dateCreated": "Jul 5, 2022 7:40:15 PM",
  "outputResources": {
    "amis": []
  },
  "accountId": "123456789012",
  "enhancedImageMetadataEnabled": true,
  "buildType": "SCHEDULED",
  "tags": {
    "internalId": "456c78b9-0e12-3f45-afb6-7e89b0f1a23b",
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image/1.0.0/7"
  }
}
```


适用于您的 Image Builder 映像的合规产品

随着安全标准的不断发展，保持合规性并保护您的组织免受网络威胁可能是一项挑战。Image Builder 集成了合规产品和 Image Builder 组件，以帮助确保您的自定义图像 AWS Marketplace 合规，并在发布商发布新版本时自动更新时保持合规。

Image Builder 可与以下合规产品集成：

- Center for Internet Security (CIS) 基准强化

  您可以使用 CIS 强化映像和相关的 CIS 强化组件来构建符合最新 CIS 基准第 1 级指导方针的自定义映像。CIS 加固镜像可在中找到。 AWS Marketplace要了解有关如何设置和使用 CIS 加固映像和强化组件的更多信息，请参阅 CIS 安全 wiki 中的 “快速入门指南：EC2 Image Builder 的 CIS 加固组件”。

> **Note**
>
> 订阅 CIS 强化映像时，您还可以访问相关的构建组件，该组件运行脚本来为您的配置强制执行 CIS 基准第 1 级指南。有关更多信息，请参阅 CIS 强化组件。

- 安全技术实施指南 (STIG)

  为了符合 STIG 标准，可以在您的 Image Builder 配方中使用亚马逊管理的 AWS Task Orchestrator and Executor (AWSTOE) STIG 组件。STIG 组件会扫描您的构建实例中是否存在错误配置，并运行修复脚本来纠正它们发现的问题。我们无法保证您使用 Image Builder 构建的映像符合 STIG 标准。您必须与贵组织的合规团队合作，以验证您的最终映像是否合规。有关可在 Image Builder 配方中使用的 AWSTOE STIG 组件的完整列表，请参阅Image Builder 的 Amazon 托管 STIG 固化组件。

在 Image Builder 中监控事件和日志

为了保持 EC2 Image Builder 管道的可靠性、可用性和性能，监控事件和日志非常重要。当 API 调用失败时，事件和日志可帮助您了解大局并深入了解细节。Image Builder 与服务集成，这些服务可以在事件符合您配置的条件时发送警报并启动自动响应。

以下主题描述您可以通过与 Image Builder 集成的服务使用的监控技术。

监控事件和日志

- 使用记录 Image Builder API 调用 CloudTrail
- 使用 Amazon 日志监控 Image Builder CloudWatch 日志

使用记录 Image Builder API 调用 CloudTrail

EC2 Image Builder 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 Image Builder 中执行的操作的记录。 CloudTrail 将 Image Builder 的所有 API 调用捕获为事件。捕获的调用包含来自 Image Builder 控制台的调用和代码对 Plans API 操作的调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 Image Builder 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 Image Builder 发出的请求、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

Image Builder 中的信息 CloudTrail

CloudTrail 在您创建账户 AWS 账户 时已在您的账户上启用。当 Image Builder 中发生活动时，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在中查看、搜索和下载最近发生的事件 AWS 账户。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的事件 AWS 账户，包括 Image Builder 的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述

- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

Image Builder 的所有操作都由 Image Builder 记录 CloudTrail 并记录在《I EC2 mage Builder API 参考》中。例如，对`CreateImagePipelineUpdateInfrastructureConfiguration`、和`StartImagePipelineExecution`操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

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

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

使用 Amazon 日志监控 Image Builder CloudWatch 日志

CloudWatch 默认情况下，日志支持处于开启状态。Image Builder 在构建过程中保留实例上的构建日志，然后在构建完成后将 CloudWatch 日志流式传输到日志。在创建最终构建映像之前，Image Builder 会从实例中删除本地日志。

镜像构建日志

映像构建日志包含单个映像构建的详细日志。Image Builder 将映像构建日志写入以下 Image Builder CloudWatch 日志组并进行直播：

LogGroup: `/aws/imagebuilder/`*`ImageName`*

LogStream (x.x.x/x): *`ImageVersion/ImageBuildVersion`*

管道执行日志

当图像管道运行或管道跳过定期运行时，Image Builder 会写入管道执行日志，以及有关发生这种情况的原因的详细信息。例如，当为计划管道配置依赖项更新时，以下场景将关联消息传送：

- Image Builder 因为没有依赖项更新而跳过管道执行时会写入日志条目。
- 当有依赖关系更新时，Image Builder 会记录已更改的资源的 ARN。

Image Builder 将管道执行日志写入以下 Image Builder CloudWatch 日志组并进行流式传输：

LogGroup: `/aws/imagebuilder/pipeline/pipeline-name`

LogStream:`2025/09/01`（YYYY/MM/DD格式为管道执行日期）

每条管道日志都会附加到当天的直播中。

自定义日志组

如果您为图像或管道日志记录指定了自定义日志组，Image Builder 会将日志写入您在管道中指定的日志组，或者如果您手动运行构建，则写入其中一个创建映像命令中指定的日志。 LogGroup 有关它在 Image Builder 中的工作原理的更多信息，请参阅配置管道日志。

选择退出

您可以通过移除与适用于您的用例的自定义角色关联的以下权限来选择退出 CloudWatch 日志流式传输。

- 日志：CreateLogGroup
- 日志：CreateLogStream
- 日志：PutLogEvents

日志类型	自定义角色
镜像构建	执行角色
管道执行	执行角色
组件日志	实例配置文件角色

对于高级故障排除，您可以使用 AWS Systems Manager Run Command 运行预定义的命令和脚本。有关更多信息，请参阅 Image Builder 问题疑难解答。

Image Builder 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云 AWS 服务 中运行的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 EC2 Image Builder 的合规性计划，请参阅 内的AWS 服务 服务合规性计划范围内的服务。
- 云端安全-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 Image Builder 时应用责任共担模型 以下主题说明如何配置 Image Builder 以实现您的安全性和合规性目标。您还将学习如何使用其他工具来监控和保护您 AWS 服务 的 Image Builder 资源。

主题

Image Builder 中的数据保护和责任 AWS 共担模型

责任共担模型 AWS 分担责任模型适用于 EC2 Image Builder 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您 AWS 服务 使用控制台、API 或与 Image Builder 或其他人合作时 AWS SDKs。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

Image Builder 中的加密和密钥管理

默认情况下，Image Builder 使用服务拥有的 KMS 密钥对传输中数据和静态数据进行加密，但以下情况除外：

- 自定义组件 – Image Builder 使用默认 KMS 密钥或服务拥有的 KMS 密钥对自定义组件进行加密。
- 映像工作流 – 如果您在工作流创建期间指定密钥，Image Builder 可以使用客户托管密钥对映像工作流进行加密。Image Builder 使用您的密钥处理加密和解密，以运行您为映像配置的工作流。

您可以通过管理自己的密钥 AWS KMS。但是，您没有权限管理 Image Builder 拥有的 Image Builder KMS 密钥。有关使用管理 KMS 密钥的更多信息 AWS Key Management Service，请参阅 AWS Key Management Service 开发人员指南中的入门。

加密上下文

为了对加密数据进行额外的完整性和真实性检查，您可以选择在加密数据时包含加密上下文。使用加密上下文对资源进行加密时，会 AWS KMS 以加密方式将上下文绑定到密文。只有当请求者为上下文提供精确、区分大小写的匹配项时，才能对资源进行解密。

本节中的策略示例使用的加密上下文类似于 Image Builder 工作流资源的Amazon 资源名称（ARN）。

使用客户托管密钥加密映像工作流

要添加一层保护，您可以使用自己的客户托管密钥对 Image Builder 工作流资源进行加密。如果您使用客户托管密钥加密您创建的 Image Builder 工作流，则必须在密钥政策中授予 Image Builder 访问权限，以便其在加密和解密工作流资源时使用您的密钥。您可以随时撤销这些访问权限。但是，如果您撤销对密钥的访问权限，Image Builder 将无法访问任何已加密的工作流。

授予 Image Builder 访问权限以使用客户托管密钥的过程分为两个步骤，如下所示：

步骤 1：为 Image Builder 工作流添加密钥政策权限

要使 Image Builder 能够在创建或使用工作流资源时对其进行加密和解密，必须在 KMS 密钥政策中指定权限。

此示例密钥政策授予了 Image Builder 管道访问权限，以在创建过程中加密工作流资源，并解密工作流资源以使用这些资源。该策略还向密钥管理员授予了访问权限。加密上下文和资源规范使用通配符来覆盖您拥有工作流资源的所有区域。

作为使用映像工作流的先决条件，您创建了一个 IAM 工作流执行角色，该角色向 Image Builder 授予运行工作流操作的权限。此处密钥政策示例中显示的第一条语句的主体必须指定您的 IAM 工作流执行角色。

有关客户托管密钥的更多信息，请参阅《AWS Key Management Service Developer Guide》中的 Managing access to customer managed keys。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "Allow access to build images with encrypted workflow",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/YourImageBuilderExecutionRole"
   },
```

```
    "Action": [
     "kms:Decrypt",
     "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
     "StringLike": {
      "kms:EncryptionContext:aws:imagebuilder:arn":
 "arn:aws:imagebuilder:*:111122223333:workflow/*"
     }
    }
   },
   {
    "Sid": "Allow access for key administrators",
    "Effect": "Allow",
    "Principal": {
     "AWS": "arn:aws:iam::111122223333:root"
    },
    "Action": [
     "kms:*"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/*"
   }
  ]
}
```

步骤 2：授予对工作流执行角色的密钥访问权限

Image Builder 为运行工作流而担任的 IAM 角色需要获得使用客户托管密钥的权限。如果无法访问您的密钥，Image Builder 将无法用其来加密或解密您的工作流资源。

编辑工作流执行角色的策略以添加以下策略语句。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "AllowAccessToWorkflowKey",
   "Effect": "Allow",
   "Action": [
```

```
    "kms:Decrypt",
    "kms:GenerateDataKey"
   ],
   "Resource": "arn:aws:kms:us-west-2:111122223333:key/key_ID",
   "Condition": {
    "StringLike": {
     "kms:EncryptionContext:aws:imagebuilder:arn":
 "arn:aws:imagebuilder:*:111122223333:workflow/*"
    }
   }
  }
 ]
}
```

AWS CloudTrail 图像工作流程的事件

以下示例显示了使用客户托管密钥存储的用于加密和解密图像工作流程的典型 AWS CloudTrail 条目。

例如： GenerateDataKey

此示例显示了 Image Builder 从 Image Builder API 操作中调用 AWS KMS GenerateDataKey API 操作时 CloudTrail 的事件可能是什么样子。CreateWorkflowImage Builder 必须先对新的工作流进行加密，然后才能创建工作流资源。

```
{
 "eventVersion": "1.08",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "PRINCIPALID1234567890:workflow-role-name",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admin/workflow-role-name",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "PRINCIPALID1234567890",
    "arn": "arn:aws:iam::111122223333:role/Admin",
    "accountId": "111122223333",
    "userName": "Admin"
   },
   "webIdFederationData": {},
   "attributes": {
```

```
    "creationDate": "2023-11-21T20:29:31Z",
    "mfaAuthenticated": "false"
   }
  },
  "invokedBy": "imagebuilder.amazonaws.com"
 },
 "eventTime": "2023-11-21T20:31:03Z",
 "eventSource": "kms.amazonaws.com",
 "eventName": "GenerateDataKey",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "imagebuilder.amazonaws.com",
 "userAgent": "imagebuilder.amazonaws.com",
 "requestParameters": {
  "encryptionContext": {
   "aws:imagebuilder:arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
sample-encrypted-workflow/1.0.0/*",
   "aws-crypto-public-key": "key value"
  },
  "keyId": "arn:aws:kms:us-west-2:111122223333:alias/ExampleKMSKey",
  "numberOfBytes": 32
 },
 "responseElements": null,
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "readOnly": true,
 "resources": [
  {
   "accountId": "111122223333",
   "type": "AWS::KMS::Key",
   "ARN": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLEzzzzz"
  }
 ],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "eventCategory": "Management"
}
```


示例：解密

此示例显示了 Image Builder 从 Image Builder API 操作中调用 AWS KMS Decrypt API 操作时 CloudTrail 的事件可能是什么样子。GetWorkflowImage Builder 管道需要先解密工作流资源，然后才能使用该资源。

```
{
 "eventVersion": "1.08",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "PRINCIPALID1234567890:workflow-role-name",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admin/workflow-role-name",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "PRINCIPALID1234567890",
    "arn": "arn:aws:iam::111122223333:role/Admin",
    "accountId": "111122223333",
    "userName": "Admin"
   },
   "webIdFederationData": {},
   "attributes": {
    "creationDate": "2023-11-21T20:29:31Z",
    "mfaAuthenticated": "false"
   }
  },
  "invokedBy": "imagebuilder.amazonaws.com"
 },
 "eventTime": "2023-11-21T20:34:25Z",
 "eventSource": "kms.amazonaws.com",
 "eventName": "Decrypt",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "imagebuilder.amazonaws.com",
 "userAgent": "imagebuilder.amazonaws.com",
 "requestParameters": {
  "keyId": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLEzzzzz",
  "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
  "encryptionContext": {
   "aws:imagebuilder:arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
sample-encrypted-workflow/1.0.0/*",
   "aws-crypto-public-key": "ABC123def4567890abc12345678/90dE/F123abcDEF+4567890abc123D
+ef1=="
  }
 },
 "responseElements": null,
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
```

```
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
 "readOnly": true,
 "resources": [
  {
   "accountId": "111122223333",
   "type": "AWS::KMS::Key",
   "ARN": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLEzzzzz"
  }
 ],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "eventCategory": "Management"
}
```


Image Builder 中的数据存储

Image Builder 不会在服务中存储您的任何日志。所有日志保存在用于生成映像的 Amazon EC2 实例上或 Systems Manager 自动化日志中。

Image Builder 中的互联网络流量隐私

通过 HTTPS 保护 Image Builder 与本地位置之间、 AWS 区域 AZs 内部以及 AWS 区域之间的连接。在账户之间没有直接连接。

Image Builder 的 Identity and Access Management 集成

主题

- 受众
- 使用身份进行身份验证
- Image Builder 如何与 IAM 策略和角色配合使用
- 在 Image Builder 中管理 S3 存储桶下载权限的数据边界
- Image Builder 基于身份的策略
- 自定义工作流程的 IAM 权限
- Image Builder 基于资源的策略
- 为 EC2 Image Builder 使用 AWS 托管策略
- 使用 Image Builder 的 IAM 服务相关角色
- Image Builder 中的 IAM 问题疑难解答

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅Image Builder 中的 IAM 问题疑难解答）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅Image Builder 如何与 IAM 策略和角色配合使用）
- IAM 管理员：编写用于管理访问权限的策略（请参阅Image Builder 基于身份的策略）

使用身份进行身份验证

有关如何为你的用户和流程提供身份验证的详细信息 AWS 账户，请参阅 IAM 用户指南中的身份。

Image Builder 如何与 IAM 策略和角色配合使用

在使用 IAM 管理对 Image Builder 的访问之前，您应该了解哪些 IAM 功能可用于 Image Builder。

要全面了解 Image Builder 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

Image Builder 的基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Image Builder 的基于身份的策略示例

要查看 Image Builder 基于身份的策略的示例，请参阅 Image Builder 基于身份的策略。

Image Builder 内基于资源的策略

支持基于资源的策略：是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

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主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Image Builder 的策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

要查看 Image Builder 操作的列表，请参阅服务授权参考中的 EC2 Image Builder 定义的操作。

Image Builder 中的策略操作在操作前使用以下前缀：

```
imagebuilder
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
 "imagebuilder:action1",
 "imagebuilder:action2"
 ]
```

要查看 Image Builder 基于身份的策略的示例，请参阅 Image Builder 基于身份的策略。

Image Builder 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 Image Builder 资源类型及其列表 ARNs，请参阅《服务授权参考》中的 EC2 Image Builder 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 EC2 Image Builder 定义的操作。

要查看 Image Builder 基于身份的策略的示例，请参阅 Image Builder 基于身份的策略。

Image Builder 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Condition` 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

有关 Image Builder 条件键的列表，请参阅服务授权参考中的 EC2 Image Builder 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 EC2 Image Builder 定义的操作。

要查看 Image Builder 基于身份的策略的示例，请参阅 Image Builder 基于身份的策略。

ACLs 在 Image Builder 中

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 Image Builder 进行 ABAC

支持 ABAC（策略中的标签）：部分支持

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`aws:RequestTag/*key-name* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 Image Builder

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

Image Builder 的跨服务主体权限

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Image Builder 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Image Builder 的功能。仅当 Image Builder 提供相关指导时才编辑服务角色。

Image Builder 的服务相关角色

支持服务关联角色：是

服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关服务相关角色的详细信息，请参阅 使用 Image Builder 的 IAM 服务相关角色。

Image Builder 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源，以及允许或拒绝操作的条件。Image Builder 支持特定的操作、资源和条件键。有关在 JSON 策略中使用的所有元素的相关信息，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

Image Builder 中的策略操作在操作前使用以下前缀：`imagebuilder:`。策略语句必须包含 `Action` 或 `NotAction` 元素。Image Builder 定义了一组自己的操作，以描述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": [
 "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 Image Builder 操作的列表，请参阅 IAM 用户指南 中的 AWS 服务的操作、资源和条件键。

使用策略管理访问

有关如何 AWS 通过创建策略并将其关联到 IAM 身份或 AWS 资源来管理中的访问权限的详细信息，请参阅 I AM 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
- EC2InstanceProfileForImageBuilderECRContainerBuilds
- AmazonSSMManagedInstanceCore

资源

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

ARN 由多个节点组成，这些节点有助于识别资源并确保名称的唯一性。名称中的最后一个节点包括资源类型、名称和 ID 格式的多种变体。Image Builder 创建资源时，它使用以下格式：

`arn:aws:imagebuilder:`*`region`*`:`*`owner`*`:`*`resource-type/resource-name/version/build-version`*

> **Note**
>
> 编译版本并不总是包含在资源 ARN 中。但是，某些 API 操作（例如 GetComponent）需要编译版本来唯一标识要检索的资源。

对于 Image Builder 在其配方中使用的资源（例如基础映像或组件），所有者节点可以是以下节点之一：

- 资源所有者的账号
- 对于 Amazon 托管资源：`aws`
- 有关 AWS Marketplace 资源：`aws-marketplace`

以下示例显示了在 Linux 上安装 Amazon CloudWatch 代理的托管组件的 ARN：

```
arn:aws:imagebuilder:us-east-1:aws:component/amazon-cloudwatch-agent-linux/1.0.1/1
```

此示例显示了来自以下内容的虚构托管组件的 ARN： AWS Marketplace

```
arn:aws:imagebuilder:us-east-1:aws-marketplace:component/example-linux-software-component/1.0.1
```

有关获取组件列表（包括使用所有权筛选器）的更多信息，请参阅列出 Image Builder 组件。

示例 ARNs

以下是您可以在 IAM 策略中指定的资源 ARNs 的一些示例：

- 实例 ARN

```
"Resource": "arn:aws:imagebuilder:us-east-1:111122223333:instance/i-1234567890abcdef0"
```

- 通配符 (*) 示例，用于指定给定账户的所有实例

```
"Resource": "arn:aws:imagebuilder:us-east-1:111122223333:instance/*"
```

- 通配符 (*) 示例，用于指定托管图像工作流程的所有版本

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:workflow/build/build-image/*"
```

- 托管映像 ARN

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-arm64/2024.12.17/1"
```

- 通配符 (*) 示例，用于指定托管映像的所有版本

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-arm64/x.x.x"
```

许多 EC2 Image Builder API 操作涉及多种资源。要在单个语句中指定多个资源，请 ARNs 用逗号分隔。

```
"Resource": [
```

```
    "resource1",
    "resource2"
]
```

条件键

Image Builder 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 全局条件上下文密钥。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。

使用此键可以根据请求中是否存在标签键来筛选访问。这允许您管理 Image Builder 创建的资源。

可用性-此密钥仅适用于`CreateInfrastrucutreConfiguration`和`UpdateInfrastructureConfiguration` APIs。

imagebuilder:/CreatedResourceTag<key>

与字符串运算符结合使用。

使用此键可按附加到 Image Builder 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 资源。

可用性-此密钥仅适用于`CreateInfrastrucutreConfiguration`和`UpdateInfrastructureConfiguration` APIs。

imagebuilder：LifecyclePolicyResourceType

与字符串运算符结合使用。

使用此密钥按请求中指定的生命周期资源类型筛选访问权限。

此键的值可以是`AMI_IMAGE`或`CONTAINER_IMAGE`。

可用性-此密钥仅适用于`CreateLifecyclePolicy`和`UpdateLifecyclePolicy`APIs。

ImageBuilder: EC2 MetadataHttpTokens

与字符串运算符结合使用。

使用此键可按请求中指定的 EC2 实例元数据 HTTP Token Requirement 来筛选访问。

此键的值可以是 `optional` 或 `required`。

可用性-此密钥仅适用于`CreateInfrastrucutreConfiguration`和`UpdateInfrastructureConfiguration` APIs。

imagebuilder：StatusTopicArn

与字符串运算符结合使用。

使用此键可按将发布终端状态通知的请求中的 SNS Topic ARN 来筛选访问。

可用性-此密钥仅适用于`CreateInfrastrucutreConfiguration`和`UpdateInfrastructureConfiguration` APIs。

示例

要查看 Image Builder 基于身份的策略的示例，请参阅 Image Builder 基于身份的策略。

Image Builder 基于资源的策略

基于资源的策略指定了指定主体可在 Image Builder 资源上执行的操作以及在什么条件下可执行。Image Builder 支持针对组件、映像和映像配方的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。您还可以使用基于资源的策略来允许 AWS 服务访问您的组件、图像和图像配方。

要了解如何将基于资源的策略附加到组件、映像或映像配方，请参阅 与共享 Image Builder 资源 AWS RAM。

> **Note**
>
> 在使用 Image Builder 更新资源策略时，更新将显示在 RAM 控制台中。

基于 Image Builder 标签的授权

您可以将标签附加到 Image Builder 资源或将请求中的标签传递到 Image Builder。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 Image Builder 资源的更多信息，请参阅 标记来自的资源 AWS CLI。

Image Builder IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的实体。

将临时凭证用于 Image Builder

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以通过调用AssumeRole或之类的 AWS STS API 操作来获取临时安全证书GetFederationToken。

服务关联角色

服务相关角色 AWS 服务 允许访问其他服务中的资源以代表您完成操作。服务关联角色显示在 IAM 账户中，并归该服务所有。具有管理员访问权限的用户可以查看但不能编辑服务相关角色的权限。

Image Builder 支持服务相关角色。有关创建或管理 Image Builder 服务相关角色的信息，请参阅 使用 Image Builder 的 IAM 服务相关角色。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在 IAM 账户中，并归该账户所有。这意味着具有管理员访问权限的用户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。

在 Image Builder 中管理 S3 存储桶下载权限的数据边界

EC2 Image Builder 维护两类 AWS 服务自有的 S3 存储桶，其中包含在您的账户中运行 Image Builder 工作负载所需的可下载资源。如果您使用数据边界来控制环境中对 Amazon S3 的访问，则可能需要明确允许访问这些存储桶。您可以使用存储桶 ARN 或存储桶 URL 将这些存储桶列入许可名单，具体取决于您如何控制对 Amazon S3 的访问权限。

组件管理引导脚本（必填）

此 S3 存储桶包含用于在用于创建映像的 EC2 实例上设置 AWSTOE 应用程序的引导脚本。Image Builder 需要下载脚本才能支持新映像的构建和测试。

- S3 存储桶 ARN：`arn:<AWS partition>:s3:::ec2imagebuilder-managed-resources-<AWS Region>-prod`
- S3 存储桶网址：`https://ec2imagebuilder-managed-resources-<AWS Region>.s3.<AWS Region>.<AWS partition-specific domain name>`

托管组件

此 S3 存储桶包含 Amazon 托管组件的包有效负载。Image Builder 需要访问权限才能下载配方中配置的所有托管组件。

- S3 存储桶 ARN：`arn:<AWS partition>:s3:::ec2imagebuilder-toe-<AWS Region>-prod`
- S3 存储桶网址：`https://ec2imagebuilder-toe-<AWS Region>.s3.<AWS Region>.<AWS partition-specific domain name>`

Image Builder 基于身份的策略

主题

- 基于身份的策略最佳实践
- 使用 Image Builder 控制台

基于身份的策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 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，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。

- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Image Builder 控制台

要访问 EC2 Image Builder 控制台，您必须拥有一组最低的权限。这些权限允许您列出和查看有关您的 AWS 账户 中的 Image Builder 资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

为确保您的 IAM 实体可以使用 Image Builder 控制台，您必须为其附加以下 AWS 托管策略之一：

- AWSImageBuilderReadOnlyAccess 策略
- AWSImageBuilderFullAccess 策略

有关 Image Builder 托管策略的更多信息，请参阅 为 EC2 Image Builder 使用 AWS 托管策略。

> **Important**
>
> AWSImageBuilderFullAccess 策略是创建 Image Builder 服务相关角色所必需的。将此策略附加到 IAM 实体时，您还必须附加以下自定义策略，并包含要使用的资源（该资源的名称中不含 `imagebuilder`）：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:*imagebuilder*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/*imagebuilder*"
        },
        {
            "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:ListBucket"
            ],
            "Resource": "arn:aws:s3::*:*imagebuilder*"
        }
    ]
}
```

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

自定义工作流程的 IAM 权限

使用带有特定步骤操作的自定义工作流程时RegisterImage，除了标准的 Image Builder 托管策略之外，可能需要其他 IAM 权限。本节介绍自定义工作流程步骤操作所需的其他权限。

RegisterImage 步骤操作权限

该RegisterImage步骤操作需要特定的 Amazon EC2 权限才能注册AMIs 和可选地检索快照标签。使用includeSnapshotTags参数时，需要额外的权限来描述快照。

RegisterImage 步骤操作所需的权限：

对于所有资源，允许执行以下操作：

- ec2:RegisterImage
- ec2:DescribeSnapshots

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:RegisterImage",
                "ec2:DescribeSnapshots"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*::image/*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "RegisterImage"
                }
            }
        }
    ]
}
```

权限详情：

- ec2:RegisterImage-需要 AMIs 从快照中注册新快照
- ec2:DescribeSnapshots-使用检索快照标签includeSnapshotTags: true以与 AMI 标签合并时必填项
- ec2:CreateTags-需要对注册的 AMI 应用标签，包括 Image Builder 默认标签和合并的快照标签

> **Note**
>
> 该`ec2:DescribeSnapshots`权限仅在`includeSnapshotTags`参数设置为时使用`true`。如果您不使用此功能，则可以省略此权限。

标签合并行为：

`includeSnapshotTags`启用后， RegisterImage 步骤操作将：

- 从块储存设备映射中指定的第一个快照中检索标签
- 排除任何 AWS 保留的标签（密钥以 “aws:” 开头的标签）
- 将快照标签与映像生成器的默认 AMI 注册标签合并
- 当标签键发生冲突时，优先使用 Image Builder 标签

Image Builder 基于资源的策略

有关如何创建组件的信息，请参阅 使用组件自定义 Image Builder 映像。

限制 Image Builder 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 Image Builder 操作。但是，请求必须来自条件中指定的 IP 地址范围。

此语句中的条件标识了允许的互联网协议版本 4 (IPv4) IP 地址的 54.240.143.* 范围，但有一个例外：54.240.143.188。

该`Condition`块使用`IpAddress`和`NotIpAddress`条件和`aws:SourceIp`条件键，后者是一个 AWS 宽范围的条件键。有关这些条件键的更多信息，请参阅在策略中指定条件。这些`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。

JSON

```
{
  "Version":"2012-10-17",
  "Id": "IBPolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
```

```
      "Effect": "Allow",
      "Action": "imagebuilder:GetComponent",
      "Resource": "arn:aws:imagebuilder:*::examplecomponent/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
         "NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}
      }
    }
  ]
}
```

为 EC2 Image Builder 使用 AWS 托管策略

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWSImageBuilderFullAccess 策略

该 AWSImageBuilderFullAccess 策略授予所附加角色对 Image Builder 资源的完全访问权限，允许该角色列出、描述、创建、更新和删除 Image Builder 资源。该策略还向相关 AWS 服务 人员授予所需的定向权限，例如验证资源或在中显示账户的当前资源 AWS 管理控制台。

权限详细信息

该策略包含以下权限：

- Image Builder – 授予管理权限，使该角色可以列出、描述、创建、更新和删除 Image Builder 资源。
- Amazon EC2 – 授予 Amazon EC2 Describe 操作的访问权限，这些操作是验证资源是否存在或获取属于账户的资源列表所必需的。
- IAM – 授予访问权限以获取和使用名称包含“imagebuilder”的实例配置文件，通过 `iam:GetRole` API 操作验证 Image Builder 服务相关角色是否存在，以及创建 Image Builder 服务相关角色。

- License Manager – 授予访问权限，以列出资源的许可证配置或许可证。
- Amazon S3 – 授予访问权限，以列出属于该账户的存储桶，以及名称中带有“imagebuilder”的 Image Builder 存储桶。
- Amazon SNS – 向 Amazon SNS 授予写入权限，以验证包含“imagebuilder”的主题的主题所有权。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 AWSImageBuilderFullAccess。

AWSImageBuilderReadOnlyAccess 策略

AWSImageBuilderReadOnlyAccess 策略提供对所有 Image Builder 资源的只读访问权限。授予权限以通过 `iam:GetRole` API 操作验证 Image Builder 服务相关角色是否存在。

权限详细信息

该策略包含以下权限：

- Image Builder – 授予对 Image Builder 资源的只读访问权限。
- IAM – 授予访问权限，以通过 `iam:GetRole` API 操作验证 Image Builder 服务相关角色是否存在。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 AWSImageBuilderReadOnlyAccess。

AWSServiceRoleForImageBuilder 策略

该AWSServiceRoleForImageBuilder政策允许 Image Builder AWS 服务 代表您致电。

权限详细信息

通过 Systems Manager 创建 Image Builder 服务相关角色时，该策略将附加到该角色。有关 Image Builder 服务相关角色的更多信息，请参阅 使用 Image Builder 的 IAM 服务相关角色。

此策略包含以下权限：

- CloudWatch 日志-授予创建 CloudWatch 日志并将其上传到名称以开头的任何日志组的访问权限`/aws/imagebuilder/`。
- Amazon EC2 — Image Builder 有权创建、拍摄快照和注册其创建的映像 (AMI)，并在您的账户中启动 EC2 实例。Image Builder 会根据需要使用相关的快照、卷、网络接口、子网、安全组、许可证配置和密钥对，前提是正在创建或使用的映像、实例和卷标有`CreatedBy: EC2 Image Builder`或`CreatedBy: EC2 Fast Launch`。

Image Builder 可以获取以下有关信息：Amazon EC2 映像、实例属性、实例状态、账户可用的实例类型、启动模板、子网、主机和 Amazon EC2 资源上的标签。

Image Builder 可以更新映像设置，以启用或禁用账户中的 Windows 实例快速启动（其中映像标有 `CreatedBy: EC2 Image Builder`）。

此外，Image Builder 可以启动、停止和终止账户中运行的实例，还可以共享 Amazon EBS 快照、创建和更新映像和启动模板，注销现有映像，添加标签，以及在您通过 Ec2ImageBuilderCrossAccountDistributionAccess 策略授予权限的账户之间复制映像。如前所述，所有这些操作都需要使用 Image Builder 标记。

- Amazon ECR – 向 Image Builder 授予访问权限，在需要进行容器映像漏洞扫描时创建存储库，并为其创建的资源添加标签，以限制其操作范围。Image Builder 还被授予访问权限，以在获取漏洞快照后删除其为扫描创建的容器映像。
- EventBridge— 授予 Image Builder 创建和管理 EventBridge 规则的权限。
- IAM – 向 Image Builder 授予访问权限，以将您账户中的任意角色传递给 Amazon EC2 和 VM Import/Export。
- Amazon Inspector – 向 Image Builder 授予访问权限，以确定 Amazon Inspector 何时完成构建实例扫描，并收集配置为允许扫描的映像的结果。
- AWS KMS：向 Amazon EBS 授予访问权限，以加密、解密或重新加密 Amazon EBS 卷。这一点非常重要，可以确保当 Image Builder 构建映像时，加密卷能够正常工作。
- License Manager – 向 Image Builder 授予访问权限，以通过 `license-manager:UpdateLicenseSpecificationsForResource` 更新 License Manager 规格。
- Amazon SNS – 向账户中的任何 Amazon SNS 主题都授予写入权限。
- Systems Manager – 向 Image Builder 授予访问权限，以列出 Systems Manager 命令及其调用、清单条目、描述实例信息和自动执行状态、描述提供实例放置支持的主机，并获取命令调用详细信息。Image Builder 还可以发送自动化信号，并停止对账户中任意资源的自动化执行。

  Image Builder 能够向标记为 `"CreatedBy": "EC2 Image Builder"` 的任何实例发出运行命令调用，用于以下脚本文件：`AWS-RunPowerShellScript`、`AWS-RunShellScript` 和 `AWSEC2-RunSysprep`。Image Builder 能够在账户中启动 Systems Manager 自动化执行，用于名称以 `ImageBuilder` 开头的自动化文档。

  Image Builder 还可以在账户中为任何实例创建或删除状态管理器关联（只要关联文档为 `AWS-GatherSoftwareInventory`），并且可以在账户中创建 Systems Manager 服务相关角色。

Image Builder 能够读取公共参数存储参数，并读取和更新前缀为的私有参数，`/imagebuilder/`这样它就可以使用 Image Builder 从新版本中创建的输出 AMI IDs 来更新参数值。

- AWS STS：向 Image Builder 授予访问权限，以将账户中名为 EC2ImageBuilderDistributionCrossAccountRole 的角色带入任何账户，前提是该角色的信任策略允许代入。这可用于跨账户映像分配。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 AWSServiceRoleForImageBuilder。

Ec2ImageBuilderCrossAccountDistributionAccess 策略

该 Ec2ImageBuilderCrossAccountDistributionAccess 策略向 Image Builder 授予权限，以在目标区域跨账户分配映像。此外，Image Builder 能够描述、复制和应用标签到账户中的任何 Amazon EC2 映像。该策略还允许通过 `ec2:ModifyImageAttribute` API 操作修改 AMI 权限。

权限详细信息

该策略包含以下权限：

- Amazon EC2 – 向 Amazon EC2 授予访问权限，以描述、复制和修改映像的属性，以及为账户中的任何 Amazon EC2 映像创建标签。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 Ec2ImageBuilderCrossAccountDistributionAccess。

EC2ImageBuilderLifecycleExecutionPolicy 策略

该EC2ImageBuilderLifecycleExecutionPolicy政策授予 Image Builder 执行诸如弃用、禁用或删除 Image Builder 图像资源及其底层资源（快照）等操作的权限AMIs，以支持映像生命周期管理任务的自动规则。

权限详细信息

该策略包含以下权限：

- Amazon EC2 — Amazon EC2 被授予访问权限，允许其对账户中标记为的亚马逊系统映像 (AMIs) 执行以下操作`CreatedBy: EC2 Image Builder`。
  - 启用和禁用 AMI。
  - 启用和禁用映像弃用。

- 描述和注销 AMI。
- 描述和修改 AMI 映像属性。
- 删除与 AMI 关联的卷快照。
- 检索资源的标签。
- 在 AMI 中添加或删除弃用标签。

- Amazon ECR – 向 Amazon ECR 授予访问权限，以对具有 `LifecycleExecutionAccess: EC2 Image Builder` 标签的 ECR 存储库上执行以下批处理操作。批处理操作支持自动化容器映像生命周期规则。
  - `ecr:BatchGetImage`
  - `ecr:BatchDeleteImage`

  在存储库级别向标记为 `LifecycleExecutionAccess: EC2 Image Builder` 的 ECR 存储库授予访问权限。
- AWS 资源组 — 授予 Image Builder 基于标签获取资源的权限。
- EC2 Image Builder – 向 Image Builder 授予权限，以删除 Image Builder 映像资源。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 EC2ImageBuilderLifecycleExecutionPolicy。

EC2InstanceProfileForImageBuilder 策略

EC2InstanceProfileForImageBuilder 策略授予 EC2 实例与 Image Builder 协同工作所需的最低权限。但这不包括使用 Systems Manager 代理所需的权限。

权限详细信息

该策略包含以下权限：

- CloudWatch 日志-授予创建 CloudWatch 日志并将其上传到名称以开头的任何日志组的访问权限`/aws/imagebuilder/`。
- Amazon EC2 — 有权描述卷和快照、创建 Image Builder 创建的卷或快照资源的快照以及为 Image Builder 资源创建标签。
- Image Builder — 授予获取任何图像生成器或 AWS Marketplace 组件的访问权限。
- AWS KMS— 如果Image Builder组件是通过加密的，则有权解密该组件。 AWS KMS
- Amazon S3 — 授予访问权限，以获取存储在名称以 `ec2imagebuilder-` ISO 开头的 Amazon S3 存储桶中的对象或文件扩展名为 ISO 的资源。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 EC2InstanceProfileForImageBuilder。

EC2InstanceProfileForImageBuilderECRContainerBuilds 策略

当 EC2 实例使用 Image Builder 构建 Docker 映像，然后在 Amazon ECR 容器存储库中注册和存储映像时，EC2InstanceProfileForImageBuilderECRContainerBuilds 策略会授予其所需的最低权限。但这不包括使用 Systems Manager 代理所需的权限。

权限详细信息

该策略包含以下权限：

- CloudWatch 日志-授予创建 CloudWatch 日志并将其上传到名称以开头的任何日志组的访问权限/aws/imagebuilder/。
- Amazon ECR – 向 Amazon ECR 授予访问权限，以获取、注册和存储容器映像，以及获取授权令牌。
- Image Builder – 授予访问权限，以获取 Image Builder 组件或容器配方。
- AWS KMS— 如果Image Builder组件或容器配方是通过加密的，则授予解密该组件或容器配方的权限。 AWS KMS
- Amazon S3 – 授予权限，以获取存储在名称以 ec2imagebuilder- 开头的 Amazon S3 存储桶中的对象。

要查看此策略的权限，请参阅《AWS 托管式策略参考》中的 EC2InstanceProfileForImageBuilderECRContainerBuilds。

Image Builder 更新 AWS 了托管策略

本节提供有关自Image Builder AWS 托管策略开始跟踪这些更改以来对该服务所做的更新的信息。有关此页面更改的自动警报，请订阅 Image Builder 文档历史记录页面上的 RSS 源。

更改	描述	日期
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改： • 已删除ssm:StartAutomationExecutio	2026 年 2 月 26 日

更改	描述	日期
	n ，因为该服务已于 2023 年完成从 SSM Automation 迁移，不再需要此权限。	
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改，以支持在配方和图像分发期间使用 AWS Systems Manager (SSM) 参数存储参数。 • 添加了 ssm: GetParameter 以允许 Image Builder 读取公共 SSM 参数和前缀为前缀的私有 SSM 参数，/imagebuilder/ 以便它们可以在配方中使用。 • 添加了 ssm: PutParameter 以允许 Image Builder 更新/imagebuilder/ 以 Image Bu IDs ilder 在新版本中创建的输出 AMI 为前缀的私有 SSM 参数。	2025 年 7 月 23 日

更改	描述	日期
EC2InstanceProfileForImageBuilder：对现有策略的更新	Image Builder 对实例配置文件策略进行了以下更改，以支持 ISO 文件下载更多文件扩展名。 • 在s3:GetObject 操作资源列表中添加了两个文件扩展名："arn:aws:s3:::*/*.iso" 和"arn:aws:s3:::*/*.Iso" 。	2025 年 5 月 19 日
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改，以支持将 Microsoft 客户端操作系统 ISO 文件作为基础映像导入。 • 添加了 ec2: RegisterImage 以允许 Image Builder 创建快照并创建和注册基准操作系统是从经过验证的 ISO 磁盘文件中导入的 AMI。	2024 年 12 月 30 日

更改	描述	日期
EC2InstanceProfileForImageBuilder：对现有策略的更新	Image Builder 对实例配置文件策略进行了以下更改，以支持从磁盘映像文件创建映像。 • 在所有资源DescribeSnapshots 上添加了以下 ec2 操作： DescribeVolumes 和，以检索详细信息。还为 Image Builder 创建的资源添加了以下操作：CreateSnapshot 卷和快照资源，以及 CreateTags。GetObject 为文件扩展名为 “ISO” 的资源添加了 s3:。	2024 年 12 月 30 日
EC2InstanceProfileForImageBuilder - 更新的策略	Image Builder 更新了`EC2InstanceProfileForImageBuilder` 政策，允许图像生成器获取 AWS Marketplace 组件。	2024 年 12 月 2 日
EC2ImageBuilderLifecycleExecutionPolicy：新策略	Image Builder 添加了包含映像生命周期管理权限的新`EC2ImageBuilderLifecycleExecutionPolicy` 策略。	2023 年 11 月 17 日

更改	描述	日期
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改以提供实例放置支持。 • 新增 ec2：DescribeHosts 允许 Image Builder 轮询主机 ID 以确定其何时处于启动实例的有效状态。 • 添加了 ssm:GetCommandInvocation，API 操作以改进 Image Builder 用于获取命令调用详细信息的方法。	2023 年 10 月 19 日
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改以提供实例放置支持。 • 新增 ec2：DescribeHosts 启用 Image Builder 轮询主机 ID 以确定其何时处于启动实例的有效状态。 • 添加了 ssm:GetCommandInvocation，API 操作以改进 Image Builder 用于获取命令调用详细信息的方法。	2023 年 9 月 28 日

更改	描述	日期
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改，以允许 Image Builder 工作流程收集 AMI 和 ECR 容器映像版本的漏洞结果。新权限支持 CVE 检测和报告功能。 • 添加了 inspector2: ListCoverage 和 inspector2: ListFindings 以允许 Image Builder 确定 Amazon Inspector 何时完成测试实例扫描，并收集配置为允许扫描的图像的结果。 • 添加了 ecr:CreateRepository，并要求 Image Builder 使用 `CreatedBy: EC2 Image Builder` (tag-on-create) 标记存储库。还添加了具有相同 CreatedBy 标签约束的 ecr:TagResource（必填 tag-on-create），以及一个要求存储库名称以开头的`image-builder-*`附加约束。名称限制可防止权限升级，并防止对 Image Builder 未创建的存储库进行更改。 • BatchDeleteImage 为带有标签的 ECR 存储库添加了 ecr:。`CreatedBy: EC2 Image Builder`此	2023 年 3 月 30 日

更改	描述	日期
	权限要求存储库名称以 `image-builder-*` 开头。 • 为 Image Builder 添加了创建和管理名称`ImageBuilder-*` 中包含的亚马逊 EventBridge 托管规则的事件权限。	
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改： • 添加了 License Manager 许可证作为 ec2: RunInstance 调用的资源 AMIs ，以允许客户使用与许可证配置关联的基础映像。	2022 年 3 月 22 日
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改： • 添加了 EC2 EnableFastLaunch API 操作的权限，以启用和禁用 Windows 实例的更快启动速度。 • 进一步缩小了 ec2：CreateTags 操作和资源标签条件的范围。	2022 年 2 月 21 日

更改	描述	日期
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 对服务角色进行了以下更改： • 已增加权限，以调用 VMIE 服务导入虚拟机并从中创建基本 AMI。 • 收紧了 ec2 的范围：CreateTags 操作和资源标签条件。	2021 年 11 月 20 日
AWSServiceRoleForImageBuilder：对现有策略的更新	Image Builder 已添加新权限，以修复多个清单关联导致映像构建卡住的问题。	2021 年 8 月 11 日
AWSImageBuilderFullAccess：对现有策略的更新	Image Builder 对完全访问权限角色进行了以下更改： • 添加允许 `ec2:DescribeInstanceTypeOfferings` 的权限。 • 添加调用 `ec2:DescribeInstanceTypeOfferings` 的权限，使 Image Builder 控制台能够准确反映账户中可用的实例类型。	2021 年 4 月 13 日
Image Builder 已开始跟踪更改	Image Builder 开始跟踪其 AWS 托管策略的更改。	2021 年 4 月 02 日

使用 Image Builder 的 IAM 服务相关角色

EC2 Image Builder 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 Image Builder 直接相关。服务相关角色由 Image Builder 预定义，包括该服务代表您呼叫他人 AWS 服务 所需的所有权限。

服务相关角色可让您更高效地设置 Image Builder，因为您不必手动添加必要的权限。Image Builder 定义其服务相关角色的权限，除非另外定义，否则只有 Image Builder 可以代入该角色。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，请参阅与 IAM 配合使用的AWS 服务，并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。请选择是与查看该服务的服务关联角色文档的链接。

Image Builder 的服务相关角色权限

Image Builder 使用AWSServiceRoleForImageBuilder服务相关角色允许 EC2 Image Builder 代表您访问 AWS 资源。服务相关角色信任 imagebuilder.amazonaws.com 服务来代入该角色。

您无需手动创建该服务相关角色。当您在 AWS 管理控制台、或 AWS API 中创建第一个 Image Builder 映像时，Image Builder 会为您创建服务相关角色。 AWS CLI

以下操作可创建新的映像：

- 在 Image Builder 控制台中运行管道向导以创建自定义映像。
- 使用以下 API 操作之一或其对应的 AWS CLI 命令：
  - CreateImageAPI 操作（create-image在 AWS CLI）。
  - ImportVmImageAPI 操作（import-vm-image在 AWS CLI）。
  - StartImagePipelineExecutionAPI 操作（start-image-pipeline-execution在 AWS CLI）。

> **Important**
>
> 如果您的帐户已删除服务相关角色，您可以使用相同的过程重新创建它。在创建第一个 EC2 Image Builder 资源时，Image Builder 将再次为您创建服务相关角色。

要查看 AWSServiceRoleForImageBuilder 的权限，请参阅 AWSServiceRoleForImageBuilder 策略 页面。要了解有关为服务相关角色配置权限的更多信息，请参阅 IAM 用户指南中的服务相关角色权限。

从您的账户中移除 Image Builder 服务相关角色

您可以使用 IAM 控制台 AWS CLI、或 AWS API 从您的账户中手动移除 Image Builder 的服务相关角色。但是，在执行此操作之前，必须确保未启用任何引用它的 Image Builder 资源。

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

清理 **AWSServiceRoleForImageBuilder** 角色使用的 Image Builder 资源

1. 在开始操作之前，请确认没有正在运行中的任何管道构建。要取消正在运行中的构建，请使用 AWS CLI中的 `cancel-image-creation` 命令。

   ```
   aws imagebuilder cancel-image-creation --image-build-version-
   arn arn:aws:imagebuilder:us-east-1:123456789012:image-pipeline/sample-pipeline
   ```

2. 将所有管道计划更改为采用手动构建流程，或者如果不再使用它们，则将其删除。有关删除资源的更多信息，请参阅 删除过期或未使用的 Image Builder 资源。

使用 IAM 删除服务相关角色

您可以使用 IAM 控制台 AWS CLI、或 AWS API 从您的账户中删除该`AWSServiceRoleForImageBuilder`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。

Image Builder 服务相关角色的受支持区域

Image Builder 支持在提供服务的所有 AWS 区域中使用服务相关角色。有关支持的 AWS 区域列表，请参阅AWS 区域和终端节点。

Image Builder 中的 IAM 问题疑难解答

主题

- 我无权在 Image Builder 中执行操作
- 我无权执行 iam：PassRole
- 我想允许我以外的人访问我 AWS 账户 的 Image Builder 资源

我无权在 Image Builder 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 imagebuilder:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 imagebuilder:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 imagebuilder:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 Image Builder。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 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此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人访问我 AWS 账户 的 Image Builder 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Image Builder 是否支持这些特征，请参阅 Image Builder 如何与 IAM 策略和角色配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Image Builder 的合规性验证

EC2 Image Builder 不在任何 AWS 合规计划的范围内。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

您可以将来自 () 的合规产品 AWS Marketplace 或 AWS Task Orchestrator and Executor (AWSTOE) 中的组件合并到 Image Builder 图片中，以帮助确保您的图片合规。有关更多信息，请参阅 适用于您的 Image Builder 映像的合规产品。

Image Builder 中的数据冗余和韧性

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。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失效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、容错能力和可扩展性。

EC2 Image Builder 服务允许您将一个区域中构建的图像与其他区域一起分发，从而为它们提供多区域弹性。 AMIs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 Image Builder 服务外部，例如，存储在 Amazon S3 存储桶中。

无法为 EC2 Image Builder 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。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

Image Builder 中的基础设施安全性

AWS 全球网络为诸如 EC2 Image Builder 之类的服务提供安全功能并控制网络访问权限。有关 AWS 为其服务提供的基础设施安全的更多信息，请参阅 AWS 安全性简介白皮书中的基础设施安全部分。

要通过 AWS 全球网络发送 Image Builder API 操作请求，您的客户端软件必须遵守以下安全准则：

- 要发送对 Image Builder API 操作的请求，客户端软件必须使用支持的传输层安全性协议 (TLS) 版本。

  > **Note**
  >
  > AWS 正在逐步取消对 TLS 版本 1.0 和 1.1 的支持。我们强烈建议您将客户端软件更新为使用 TLS 版本 1.2 或更高版本，这样您可以继续保持连接。有关更多信息，请参阅此 AWS 安全博客文章。

- 客户端软件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)。大多数当前系统（如 Java 7 及更高版本）都支持这些模式。
- 您必须使用访问密钥 ID 和与 AWS Identity and Access Management (IAM) 委托人关联的私有访问密钥来签署 API 请求。或者，您可以为您的请求使用 AWS Security Token Service (AWS STS) 生成临时安全凭证。

此外，Image Builder 用于构建和测试映像的 EC2 实例必须具有访问 AWS Systems Manager的权限。

Image Builder 映像的补丁管理

AWS 提供 AMIs 每月管理的更新，其中包含适用于以下操作系统的最新更新和安全补丁。您可以将它们 AMIs 用作自定义的基本图片。有关更多信息，请参阅 支持的操作系统。

- Linux 发行版包括亚马逊 Linux 2 AL2023、红帽企业 Linux (RHEL)、CentOS、Ubuntu、SUSE Linux 企业服务器
- Windows Server 2016 及更高版本

- macOS 10.14.x 及更高版本

根据责任共担模式，在创建自定义映像后，您需要负责修补 Amazon EC2 系统。如果可以轻松替换您应用程序工作负载中的 EC2 实例，则更新基础 AMI 并根据该映像重新部署所有计算节点可能会非常高效。

> **Note**
>
> 要进行 macOS 修补，我们建议您创建新版本的配方，使用最新的托管 AMI 作为基础映像，然后通过配方和其他映像构建资源构建更新的自定义映像。如果您的 Mac 实例不能轻松替换，请参阅《Amazon EC2 用户指南》中的更新 Mac 实例上的操作系统和软件页面来了解更多信息。

您可以通过以下两种方式让 Image Builder 保持 AMIs 最新状态。

- AWS提供的修补组件 – EC2 Image Builder 提供以下构建组件，它们安装所有待处理的操作系统更新：
  - `update-linux`
  - `update-windows`

  这些组件使用 `UpdateOS` 操作模块。有关更多信息，请参阅 UpdateOS。通过从 AWS提供的组件列表中选择组件，可以将组件添加到您的映像构建管道中。
- 带有修补操作的自定义构建组件-要有选择地在支持的操作系统上安装或更新补丁 AMIs，您可以创作 Image Builder 组件来安装所需的补丁。自定义组件可以使用 shell 脚本（Bash 或 PowerShell）安装补丁，也可以使用`UpdateOS`操作模块来指定要安装或排除的修补程序。有关更多信息，请参阅 AWSTOE 组件管理器支持的操作模块。

  使用 `UpdateOS` 操作模块的组件（仅限 Linux 和 Windows。macOS 不支持 `UpdateOS` 操作模块。）

```
schemaVersion: 1.0
phases:
  - name: build
 steps:
   - name: UpdateOS
  action: UpdateOS
```

使用 Bash 安装 yum 更新的组件

```
schemaVersion: 1.0
phases:
  - name: build
 steps:
   - name: InstallYumUpdates
  action: ExecuteBash
  inputs:
    commands:
   - sudo yum update -y
```

Image Builder 的安全最佳实践

EC2 Image Builder 提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合环境或不满足环境要求，请将其视为有用的考虑因素而不是惯例。

- 不要在 Image Builder 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。
- 定期将托管 AMI 更新应用于您的 macOS 配方，并创建新映像以启动安装了最新安全补丁的实例。

我们强烈建议您测试映像，以验证安全状况和适用的安全合规性级别。Amazon Inspector 等此类解决方案可以帮助验证映像的安全和合规性状况。

IMDSv2 适用于 Image Builder 管道

当您的 Image Builder 管道运行时，它会发送 HTTP 请求以启动 EC2 实例，Image Builder 使用这些实例来构建和测试您的映像。要配置您的管道用于启动请求的 IMDS 版本，请在 Image Builder 基础架构配置实例元数据设置中设置 `httpTokens` 参数。

> **Note**
>
> 我们建议您将 Image Builder 从管道构建中启动的所有 EC2 实例配置为使用， IMDSv2 以便实例元数据检索请求需要签名的令牌标头。

有关 Image Builder 基础设施配置的更多信息，请参阅 管理 Image Builder 基础设施配置 有关 Linux 映像的 EC2 实例元数据选项的更多信息，请参阅《Amazon EC2 用户指南》中的配置实例元数据服务选项。有关 Windows 映像，请参阅《Amazon EC2 用户指南》中的配置实例元数据服务选项。

需要在构建后进行清理

Image Builder 完成自定义映像的所有构建步骤后，Image Builder 会为测试和映像创建准备构建实例。在关闭构建实例以创建快照之前，Image Builder 会执行以下清理操作以确保映像的安全：

Linux

Image Builder 管道运行清理脚本，以帮助确保最终映像遵循安全最佳实践，并删除任何不应延续到快照中的构建构件或设置。但是，您可以跳过脚本的各个部分，或者完全覆盖用户数据。因此，Image Builder 管道生成的映像不一定符合任何特定的监管标准。

当管道完成其构建期和测试期后，Image Builder 会在创建输出映像之前自动运行以下清理脚本。

> **Important**
>
> 如果您在配方中覆盖用户数据，则脚本将无法运行。在这种情况下，请确保在用户数据中包含一个用于创建名为 `perform_cleanup` 的空文件的命令。Image Builder 会检测到此文件并在创建新映像之前运行清理脚本。

```
#!/bin/bash
if [[ ! -f {{workingDirectory}}/perform_cleanup ]]; then
    echo "Skipping cleanup"
    exit 0
else
    sudo rm -f {{workingDirectory}}/perform_cleanup
fi

function cleanup() {
    FILES=("$@")
```

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


# Clean up for cloud-init files
CLOUD_INIT_FILES=(
    "/etc/sudoers.d/90-cloud-init-users"
    "/etc/locale.conf"
    "/var/log/cloud-init.log"
    "/var/log/cloud-init-output.log"
)
if [[ -f {{workingDirectory}}/skip_cleanup_cloudinit_files ]]; then
    echo "Skipping cleanup of cloud init files"
else
    echo "Cleaning up cloud init files"
    cleanup "${CLOUD_INIT_FILES[@]}"
    if [[ $( sudo find /var/lib/cloud -type f | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting files within /var/lib/cloud/*"
        sudo find /var/lib/cloud -type f -exec shred -zuf {} \;
    fi;

    if [[ $( sudo ls /var/lib/cloud | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/lib/cloud/*"
        sudo rm -rf /var/lib/cloud/* || true
    fi;
fi;


# Clean up for temporary instance files
INSTANCE_FILES=(
    "/etc/.updated"
    "/etc/aliases.db"
    "/etc/hostname"
    "/var/lib/misc/postfix.aliasesdb-stamp"
    "/var/lib/postfix/master.lock"
```

```
    "/var/spool/postfix/pid/master.pid"
    "/var/.updated"
    "/var/cache/yum/x86_64/2/.gpgkeyschecked.yum"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_files ]]; then
    echo "Skipping cleanup of instance files"
else
    echo "Cleaning up instance files"
    cleanup "${INSTANCE_FILES[@]}"
fi;


# Clean up for ssh files
SSH_FILES=(
    "/etc/ssh/ssh_host_rsa_key"
    "/etc/ssh/ssh_host_rsa_key.pub"
    "/etc/ssh/ssh_host_ecdsa_key"
    "/etc/ssh/ssh_host_ecdsa_key.pub"
    "/etc/ssh/ssh_host_ed25519_key"
    "/etc/ssh/ssh_host_ed25519_key.pub"
    "/root/.ssh/authorized_keys"
)
if [[ -f {{workingDirectory}}/skip_cleanup_ssh_files ]]; then
    echo "Skipping cleanup of ssh files"
else
    echo "Cleaning up ssh files"
    cleanup "${SSH_FILES[@]}"
    USERS=$(ls /home/)
    for user in $USERS; do
        echo Deleting /home/"$user"/.ssh/authorized_keys;
        sudo find /home/"$user"/.ssh/authorized_keys -type f -exec shred -zuf {} \;
    done
    for user in $USERS; do
        if [[ -f /home/"$user"/.ssh/authorized_keys ]]; then
            echo Failed to delete /home/"$user"/.ssh/authorized_keys;
            exit 1
        fi;
    done;
fi;


# Clean up for instance log files
INSTANCE_LOG_FILES=(
    "/var/log/audit/audit.log"
```

```
    "/var/log/boot.log"
    "/var/log/dmesg"
    "/var/log/cron"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_log_files ]]; then
    echo "Skipping cleanup of instance log files"
else
    echo "Cleaning up instance log files"
    cleanup "${INSTANCE_LOG_FILES[@]}"
fi;

# Clean up for TOE files
if [[ -f {{workingDirectory}}/skip_cleanup_toe_files ]]; then
    echo "Skipping cleanup of TOE files"
else
    echo "Cleaning TOE files"
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]];
 then
        echo "Deleting files within {{workingDirectory}}/TOE_*"
        sudo find {{workingDirectory}}/TOE_* -type f -exec shred -zuf {} \;
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]];
 then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]];
 then
        echo "Deleting {{workingDirectory}}/TOE_*"
        sudo rm -rf {{workingDirectory}}/TOE_*
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]];
 then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
    fi
fi

# Clean up for ssm log files
if [[ -f {{workingDirectory}}/skip_cleanup_ssm_log_files ]]; then
    echo "Skipping cleanup of ssm log files"
else
    echo "Cleaning up ssm log files"
    if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
```

```
        echo "Deleting files within /var/log/amazon/ssm/*"
        sudo find /var/log/amazon/ssm -type f -exec shred -zuf {} \;
    fi
    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
    if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Deleting /var/log/amazon/ssm/*"
        sudo rm -rf /var/log/amazon/ssm
    fi
    if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
    fi
fi


if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
    echo "Deleting /var/log/sa/sa*"
    sudo shred -zuf /var/log/sa/sa*
fi
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

if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
        echo "Deleting /var/lib/dhclient/dhclient*.lease"
        sudo shred -zuf /var/lib/dhclient/dhclient*.lease
fi
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

if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Deleting files within /var/tmp/*"
        sudo find /var/tmp -type f -exec shred -zuf {} \;
fi
if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/tmp"
```

```
        exit 1
fi
if [[ $( sudo ls /var/tmp | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/tmp/*"
        sudo rm -rf /var/tmp/*
fi

# Shredding is not guaranteed to work well on rolling logs

if [[ -f "/var/lib/rsyslog/imjournal.state" ]]; then
        echo "Deleting /var/lib/rsyslog/imjournal.state"
        sudo shred -zuf /var/lib/rsyslog/imjournal.state
        sudo rm -f /var/lib/rsyslog/imjournal.state
fi

if [[ $( sudo ls /var/log/journal/ | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/log/journal/*"
        sudo find /var/log/journal/ -type f -exec shred -zuf {} \;
        sudo rm -rf /var/log/journal/*
fi

sudo touch /etc/machine-id
```

Windows

Image Builder 管道在自定义 Windows 映像后，将运行 Microsoft Sysprep 实用程序。 这些操作遵循了强化和清理图像AWS 的最佳实践。

macOS

Image Builder 管道运行清理脚本，以帮助确保最终映像遵循安全最佳实践，并删除任何不应延续到快照中的构建构件或设置。但是，您可以跳过脚本的各个部分，或者完全覆盖用户数据。因此，Image Builder 管道生成的映像不一定符合任何特定的监管标准。

当管道完成其构建期和测试期后，Image Builder 会在创建输出映像之前自动运行以下清理脚本。

> **Important**
>
> 如果您在配方中覆盖用户数据，则脚本将无法运行。在这种情况下，请确保在用户数据中包含一个用于创建名为 `perform_cleanup` 的空文件的命令。Image Builder 会检测到此文件并在创建新映像之前运行清理脚本。

```
#!/bin/bash
if [[ ! -f {{workingDirectory}}/perform_cleanup ]]; then
  echo "Skipping cleanup"
  exit 0
else
  sudo rm -f {{workingDirectory}}/perform_cleanup
fi

function cleanup() {
  FILES=("$@")
  for FILE in "${FILES[@]}"; do
      if [[ -f "$FILE" ]]; then
          echo "Deleting $FILE";
          sudo rm -f $FILE;
      fi;
      if [[ -f $FILE ]]; then
          echo "Failed to delete '$FILE'. Failing."
          exit 1
      fi;
  done
};

# Clean up for cloud-init files
CLOUD_INIT_FILES=(
  "/etc/sudoers.d/90-cloud-init-users"
  "/etc/locale.conf"
  "/var/log/cloud-init.log"
  "/var/log/cloud-init-output.log"
)
if [[ -f {{workingDirectory}}/skip_cleanup_cloudinit_files ]]; then
  echo "Skipping cleanup of cloud init files"
else
  echo "Cleaning up cloud init files"
  cleanup "${CLOUD_INIT_FILES[@]}"
  if [[ $( sudo find /var/lib/cloud -type f | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting files within /var/lib/cloud/*"
      sudo find /var/lib/cloud -type f -exec rm -f {} \;
  fi;

  if [[ $( sudo ls /var/lib/cloud | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting /var/lib/cloud/*"
      sudo rm -rf /var/lib/cloud/* || true
  fi;
```

```
fi;


# Clean up for temporary instance files
INSTANCE_FILES=(
  "/etc/.updated"
  "/etc/aliases.db"
  "/etc/hostname"
  "/var/lib/misc/postfix.aliasesdb-stamp"
  "/var/lib/postfix/master.lock"
  "/var/spool/postfix/pid/master.pid"
  "/var/.updated"
  "/var/cache/yum/x86_64/2/.gpgkeyschecked.yum"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_files ]]; then
  echo "Skipping cleanup of instance files"
else
  echo "Cleaning up instance files"
  cleanup "${INSTANCE_FILES[@]}"
fi;


# Clean up for ssh files
SSH_FILES=(
  "/etc/ssh/ssh_host_rsa_key"
  "/etc/ssh/ssh_host_rsa_key.pub"
  "/etc/ssh/ssh_host_ecdsa_key"
  "/etc/ssh/ssh_host_ecdsa_key.pub"
  "/etc/ssh/ssh_host_ed25519_key"
  "/etc/ssh/ssh_host_ed25519_key.pub"
  "/root/.ssh/authorized_keys"
)
if [[ -f {{workingDirectory}}/skip_cleanup_ssh_files ]]; then
  echo "Skipping cleanup of ssh files"
else
  echo "Cleaning up ssh files"
  cleanup "${SSH_FILES[@]}"
  USERS=$(ls /home/)
  for user in $USERS; do
      echo Deleting /home/"$user"/.ssh/authorized_keys;
      sudo find /home/"$user"/.ssh/authorized_keys -type f -exec rm -f {} \;
  done
  for user in $USERS; do
      if [[ -f /home/"$user"/.ssh/authorized_keys ]]; then
```

```
          echo Failed to delete /home/"$user"/.ssh/authorized_keys;
          exit 1
      fi;
  done;
fi;


# Clean up for instance log files
INSTANCE_LOG_FILES=(
  "/var/log/audit/audit.log"
  "/var/log/boot.log"
  "/var/log/dmesg"
  "/var/log/cron"
  "/var/log/amazon/ec2/ec2-macos-init.log"
  "/var/log/amazon/ec2/ena-ethernet.log"
  "/var/log/amazon/ec2/system-monitoring.log"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_log_files ]]; then
  echo "Skipping cleanup of instance log files"
else
  echo "Cleaning up instance log files"
  cleanup "${INSTANCE_LOG_FILES[@]}"
fi;

# Clean up for TOE files
if [[ -f {{workingDirectory}}/skip_cleanup_toe_files ]]; then
  echo "Skipping cleanup of TOE files"
else
  echo "Cleaning TOE files"
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]]; then
      echo "Deleting files within {{workingDirectory}}/TOE_*"
      sudo find {{workingDirectory}}/TOE_* -type f -exec rm -f {} \;
  fi
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]]; then
      echo "Failed to delete {{workingDirectory}}/TOE_*"
      exit 1
  fi
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]]; then
      echo "Deleting {{workingDirectory}}/TOE_*"
      sudo rm -rf {{workingDirectory}}/TOE_*
  fi
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]]; then
      echo "Failed to delete {{workingDirectory}}/TOE_*"
      exit 1
```

```
  fi
fi

# Clean up for ssm log files
if [[ -f {{workingDirectory}}/skip_cleanup_ssm_log_files ]]; then
  echo "Skipping cleanup of ssm log files"
else
  echo "Cleaning up ssm log files"
  if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
      echo "Deleting files within /var/log/amazon/ssm/*"
      sudo find /var/log/amazon/ssm -type f -exec rm -f {} \;
  fi
  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
  if [[ -d "/var/log/amazon/ssm" ]]; then
      echo "Deleting /var/log/amazon/ssm/*"
      sudo rm -rf /var/log/amazon/ssm
  fi
  if [[ -d "/var/log/amazon/ssm" ]]; then
      echo "Failed to delete /var/log/amazon/ssm"
      exit 1
  fi
fi


if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
  echo "Deleting /var/log/sa/sa*"
  sudo rm -f /var/log/sa/sa*
fi
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

if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
      echo "Deleting /var/lib/dhclient/dhclient*.lease"
      sudo rm -f /var/lib/dhclient/dhclient*.lease
fi
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

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

# Shredding is not guaranteed to work well on rolling logs

if [[ -f "/var/lib/rsyslog/imjournal.state" ]]; then
      echo "Deleting /var/lib/rsyslog/imjournal.state"
      sudo rm -f /var/lib/rsyslog/imjournal.state
      sudo rm -f /var/lib/rsyslog/imjournal.state
fi

if [[ $( sudo ls /var/log/journal/ | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting /var/log/journal/*"
      sudo find /var/log/journal/ -type f -exec rm -f {} \;
      sudo rm -rf /var/log/journal/*
fi

sudo touch /etc/machine-id
```

覆盖 Linux 清理脚本

Image Builder 创建的映像在默认情况下是安全的，并遵循我们的安全最佳实践。但是，一些更高级的使用案例可能需要您跳过内置清理脚本的一个或多个部分。如果您确实需要跳过一些清理部分，我们强烈建议您测试输出 AMI，以确保映像的安全。

Important

跳过清理脚本中的一些部分可能会导致敏感信息（例如所有者帐户详细信息或 SSH 密钥）被包含在最终映像中，并被包含在从该映像启动的任何实例中。在不同的可用区、区域或账户中启动时，您也可能会遇到问题。

下表概述了清理脚本的各个部分，这些部分中删除的文件，以及可用于标记 Image Builder 应跳过的某个部分的文件名。要跳过清理脚本的特定部分，您可以使用 CreateFile 组件操作模块或用户数据中的命令（如果已覆盖）来创建一个空文件，其名称在跳过部分文件名列中指定。

Note

为跳过清理脚本的某一部分而创建的文件不应包含文件扩展名。例如，如果您想跳过脚本的 `CLOUD_INIT_FILES` 部分，但创建了一个名为 `skip_cleanup_cloudinit_files.txt` 的文件，那么 Image Builder 将无法识别要跳过的文件。

Input

清理部分	文件已删除	跳过部分文件名

`CLOUD_INIT_FILES`	`/etc/sudoers.d/90-cloud-init-users`
	`/etc/locale.conf`
	`/var/log/cloud-init.log`
	`/var/log/cloud-init-output.log`
		`skip_cleanup_cloudinit_files`

`INSTANCE_FILES`	`/etc/.updated`
	`/etc/aliases.db`
	`/etc/hostname`
		`skip_cleanup_instance_files`

清理部分	文件已删除	跳过部分文件名
	/var/lib/misc/postfix.aliasesdb-stamp /var/lib/postfix/master.lock /var/spool/postfix/pid/master.pid /var/.updated /var/cache/yum/x86_64/2/.gpgkeyschecked.yum	
SSH_FILES	/etc/ssh/ssh_host_rsa_key /etc/ssh/ssh_host_rsa_key.pub /etc/ssh/ssh_host_ecdsa_key /etc/ssh/ssh_host_ecdsa_key.pub /etc/ssh/ssh_host_ed25519_key /etc/ssh/ssh_host_ed25519_key.pub /root/.ssh/authorized_keys /home/<all users>/.ssh/authorized_keys;	skip_cleanup_ssh_files

清理部分	文件已删除	跳过部分文件名
INSTANCE_LOG_FILES	/var/log/audit/audit.log /var/log/boot.log /var/log/dmesg /var/log/cron	skip_cleanup_instance_log_files
TOE_FILES	{{workingDirectory}}/TOE_*	skip_cleanup_toe_files
SSM_LOG_FILES	/var/log/amazon/ssm/*	skip_cleanup_ssm_log_files

Image Builder 问题疑难解答

EC2 Image Builder 与集成， AWS 服务 用于监控和故障排除，可帮助您解决映像构建问题。Image Builder 跟踪并显示映像构建过程中每个步骤的进度。此外，Image Builder 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 Run Command 运行预定义的命令和脚本。

内容

管道构建故障排除

如果 Image Builder 管道构建失败，Image Builder 将返回一条描述此失败的错误消息。Image Builder 还会在失败消息中返回 `workflow execution ID`，例如下面的输出示例：

```
Workflow Execution ID: wf-12345abc-6789-0123-abc4-567890123abc failed with reason: …
```

Image Builder 通过一系列步骤来安排和指导映像构建操作，这些步骤是为其标准映像创建过程中的运行时阶段定义的。每个流程的构建和测试阶段都有一个关联的工作流程。当 Image Builder 运行工作流程来构建或测试新映像时，它会生成一个用于跟踪运行时详细信息的工作流程元数据资源。

容器映像还有一个在分发期间运行的额外工作流程。

研究工作流程运行时实例故障的详细信息

要解决工作流程的运行时故障，您可以使用调用GetWorkflowExecution和 ListWorkflowStepExecutionsAPI 操作`workflow execution ID`。

查看工作流程运行时日志

- 亚马逊 CloudWatch 日志

  Image Builder 将详细的工作流程执行日志发布到以下 Image Builder CloudWatch 日志组和直播：

  使用 CloudWatch Logs，您可以使用筛选模式搜索日志数据。有关更多信息，请参阅 Amazon Logs 用户指南中的使用筛选模式搜索 CloudWatch 日志数据。

- AWS CloudTrail

  CloudTrail 如果在您的账户中激活了所有构建活动，则所有构建活动也会被登录。您可以按来源筛选 CloudTrail 事件`imagebuilder.amazonaws.com`。或者，您可以搜索执行日志中返回的 Amazon EC2 实例 ID，以查看有关管道执行的更多详细信息。

- Amazon Simple Storage Service (S3)

  如果您在基础设施配置中指定了 S3 存储桶名称和键前缀，则工作流程步骤运行时日志路径将遵循以下模式：

  ```
  S3://S3BucketName/KeyPrefix/ImageName/ImageVersion/ImageBuildVersion/
  WorkflowExecutionId/StepName
  ```

  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、已运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。

默认情况下，当管道出现故障时，Image Builder 会关闭正在运行的 Amazon EC2 构建或测试实例。您可以更改管道使用的基础设施配置资源的实例设置，以保留您的构建或测试实例以进行故障排除。

要在控制台中更改实例设置，您必须清除基础设施配置资源故障排除设置部分中的失败时终止实例复选框。

您也可以在 AWS CLI中使用 update-infrastructure-configuration 命令更改实例设置。在 JSON 文件将 `terminateInstanceOnFailure` 值设置为 `false`，该命令通过 `--cli-input-json` 参数引用该值。有关更多信息，请参阅 更新基础设施配置。

故障排除场景

本节列出了以下详细的故障排除场景：

- 访问被拒绝 - 状态码 403
- 验证构建实例上的 Systems Manager 代理可用性时构建超时
- Windows 辅助磁盘在启动时处于离线状态
- 使用 CIS 强化的基础映像构建失败
- AssertInventoryCollection 失败（Systems Manager 自动化）

要查看场景的详细信息，请选择场景标题将其展开。您可以同时扩展多个标题。

访问被拒绝 - 状态码 403

说明

管道构建失败，显示 “AccessDenied：访问被拒绝” 状态代码：403"。

原因

可能的原因包括：

- 实例配置文件没有访问APIs 或组件资源所需的权限。
- 实例配置文件角色缺少登录 Amazon S3 所需的权限。最常见的是，当实例配置文件角色对您的 S3 存储桶没有PutObject权限时，就会发生这种情况。

解决方案

根据原因，可以按以下操作解决此问题：

- 实例配置文件缺少托管策略 – 将缺少的策略添加到您的实例配置文件角色中。然后再次运行管道。
- 实例配置文件缺少 S3 存储桶的写入权限 — 向您的实例配置文件角色添加授予写入您的 S3 存储桶的PutObject权限的策略。然后再次运行管道。

验证构建实例上的 Systems Manager 代理可用性时构建超时

说明

管道构建失败，显示 “状态 = TimedOut '” 和 “失败消息 = '步骤验证目标实例上的 Systems Manager 代理可用性时步骤超时'”。

原因

可能的原因包括：

- 启动以执行构建操作和运行组件的实例无法访问 Systems Manager 端点。
- 实例配置文件没有所需的权限。

解决方案

根据可能的原因，可以按以下操作解决此问题：

- 访问问题，私有子网 — 如果您在私有子网中构建，请确保已为 Systems Manager、Image Builder 和 Amazon S3CloudWatch/（如果要记录）设置了PrivateLink 终端节点。有关设置 PrivateLink 终端节点的更多信息，请参阅通过访问 AWS 服务 AWS PrivateLink。
- 缺少权限 - 将以下托管策略添加到 Image Builder 的 IAM 服务相关角色中：
  - EC2InstanceProfileForImageBuilder
  - EC2InstanceProfileForImageBuilderECRContainer构建
  - 亚马逊 SSMManaged InstanceCore

  有关 Image Builder 服务相关角色的更多信息，请参阅 使用 Image Builder 的 IAM 服务相关角色。

Windows 辅助磁盘在启动时处于离线状态

说明

当用于构建 Image Builder Windows AMI 的实例类型与用于从 AMI 启动的实例类型不匹配时，可能会出现非根卷在启动时处于离线状态的问题。这主要发生在构建实例使用的架构比启动实例更新的架构时。

以下示例演示了在 EC2 Nitro 实例类型上构建并在 EC2 Xen 实例上启动的 Image Builder AMI 时会发生什么：

构建实例类型：m5.large (Nitro)

启动实例类型：t2.medium (Xen)

```
PS C:\Users\Administrator> get-disk
Number  Friendly Name  Serial Number         Health Status  Operational Status  Total
 Size  Partition Style
------  -------------  -------------         -------------  ------------------
 ----------  ---------------
0       AWS PVDISK     vol0abc12d34e567f8a9  Healthy        Online                  30
 GB  MBR
1       AWS PVDISK     vol1bcd23e45f678a9b0  Healthy        Offline                  8
 GB  MBR
```

原因

由于 Windows 的默认设置，新发现的磁盘不会自动联机并进行格式化。在 EC2 上实例类型更改时，Windows 会将其视为发现的新磁盘。这是因为底层驱动程序发生了变化。

解决方案

我们建议您在构建要启动的 Windows AMI 时使用相同的实例类型系统。请勿在基础设施配置中包含基于不同系统构建的实例类型。如果您指定的任何实例类型使用 Nitro 系统，则它们都应使用 Nitro 系统。

有关基于 Nitro 系统构建的实例的更多信息，请参阅《Amazon EC2 用户指南》中的基于 Nitro 系统构建实例。

使用 CIS 强化的基础映像构建失败

说明

您使用的是 CIS 强化的基础映像，但构建失败。

原因

当 `/tmp` 目录被归类为 `noexec` 时，可能会导致 Image Builder 出现故障。

解决方案

在映像配方 `workingDirectory` 字段中为工作目录选择其他位置。有关更多信息，请参阅ImageRecipe数据类型说明。

AssertInventoryCollection 失败（Systems Manager 自动化）

说明

Systems Manager Automation 显示在 `AssertInventoryCollection` 自动化步骤中的一个失败。

原因

您或您的组织可能已创建一个 Systems Manager State Manager 关联，用于收集 EC2 实例的清单信息。如果您的 Image Builder 管道启用了增强型映像元数据收集（这是默认设置），Image Builder 会尝试为构建实例创建新的清单关联。但是，Systems Manager 不允许托管实例存在多个清单关联，如果已经存在一个关联，则会阻止产生新关联。这会导致操作失败，并导致管道构建失败。

解决方案

要解决此问题，请使用以下其中一种方法，关闭增强型映像元数据收集：

- 在控制台中更新您的映像管道，以清除“启用增强型元数据收集”复选框。保存更改并运行管道构建。

有关使用 EC2 Image Builder 控制台更新 AMI 映像管道的更多信息，请参阅 通过控制台更新 AMI 映像管道。有关使用 EC2 Image Builder 控制台更新容器映像管道的更多信息，请参阅 通过控制台更新容器映像管道。

- 您也可以在 AWS CLI中使用 update-image-pipeline 命令更新映像管道。为此，请在您的 JSON 文件中包含 EnhancedImageMetadataEnabled 属性，将其设置为 false。以下示例显示属性设置为 false。

```
{
    "name": "MyWindows2019Pipeline",
    "description": "Builds Windows 2019 Images",
    "enhancedImageMetadataEnabled": false,
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2020.12.03",
    "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"
}
```

为防止新管道发生这种情况，请在使用 EC2 Image Builder 控制台创建新管道时，清除“启用增强型元数据收集”复选框，或者在使用 AWS CLI创建管道时，false 在 JSON 文件中设置 EnhancedImageMetadataEnabled 属性的值。

Image Builder 用户指南变更文档历史记录

下表按日期顺序描述文档的重要更改。要获得本文档的更新通知，您可以订阅 RSS 源。

- API 版本：2025-04-30

变更	说明	日期
增加了对 Windows Server 2025 的支持	增加了对Windows Server 2025 STIG版本的支持，并STIGs 适用于2026年第一季度发布的所有合规级别（ ）low/medium/high。	2026 年 3 月 10 日
IAM 政策更新：服务角色策略	已`ssm:StartAutomationExecution` 从服务相关角色策略中移除，因为该服务已于 2023 年完成从 SSM Automation 迁移。有关更多信息，请参阅 AWSServiceRoleForImageBuilder 策略。	2026 年 2 月 26 日
功能发布：生命周期策略增强	生命周期策略增强功能包括通过使用服务默认值创建基于控制台的角色来简化角色管理、基于配方的策略（1.0.x、1.x.x、x.x.x）的通配符语义版本支持以单个策略定位多个配方版本、生命周期策略的基于标签的资源集合，以及控制台支持 AMI 和基于标签的策略上的排除规则。	2026 年 2 月 26 日
图像分发增强功能	增强功能包括能够直接执行分发活动，而无需重新运行完整的映像构建。	2025 年 11 月 20 日

自动版本控制和 IaC 增强功能	现在，自动版本递增和通配符版本引用消除了对配方、组件和工作流程的手动版本管理。通过控制台中的试运行和增强的创作功能获得组件测试体验。	2025 年 11 月 20 日
图像工作流程支持 Lambda 和 Step Functions	图像工作流程现在支持调用 Lambda 函数和执行 Step Functions 状态机，从而在图像创建中实现复杂的多步骤工作流程和自定义验证逻辑。	2025 年 11 月 13 日
图像管道增强功能	增强功能包括支持管道执行日志、可配置的 CloudWatch 日志组，以及自动禁用重复失败的定时管道的配置。其他更新包括扩展的管道计划信息和更新的 CloudWatch 日志事件消息示例。	2025 年 9 月 29 日
2023 年亚马逊 Linux 的新脚本	增加了对 SUSE Linux 企业服务器 (SLES) 操作系统的支持。STIG版本和2025年第三季度发布的STIG版本适用于所有合规级别（ ）low/medium/high。	2025 年 9 月 4 日
2023 年亚马逊 Linux 的新脚本	添加了专门为亚马逊 Linux 2023 量身定制的新脚本。	2025 年 8 月 26 日
IAM 政策更新：服务角色策略	更新了服务相关角色策略，以支持在配方和图像分发期间使用 SSM 参数。有关更多信息，请参阅 AWSServiceRoleForImageBuilder 策略。	2025 年 7 月 23 日

STIG 2025 年第二季度更新	更新了 Windows STIG 版本并应用了 2025 年第二季度发布的 STIGS。	2025 年 6 月 26 日
IAM 政策更新：实例配置文件策略	添加`.iso`和`.Iso`作为有效的文件扩展名，以便从 Amazon S3 下载 ISO 磁盘映像。有关更多信息，请参阅 EC2InstanceProfileForImageBuilder 策略。	2025 年 5 月 19 日
STIG 2025 年第一季度更新	更新了 Windows STIG 版本，并应用了 2025 年第一季度发布的 STIGS 和 SCAP 组件版本。	2025 年 5 月 5 日
功能发布：Support SSM 参数	Image Builder 现在支持在配方和图像分发期间使用 AWS Systems Manager(SSM) 参数存储参数。	2025 年 4 月 30 日
STIG 第四季度更新	更新了 Windows STIG 版本并应用了 2024 年第四季度发布的 STIGS。	2025 年 2 月 4 日
IAM 政策更新：服务角色策略	更新了服务相关角色策略，以支持从导入的官方 Microsoft 客户端操作系统 ISO 文件中创建映像。有关更多信息，请参阅 AWSServiceRoleForImageBuilder 策略。	2024 年 12 月 30 日
IAM 政策更新：实例配置文件策略	更新了实例配置文件角色策略以支持从磁盘映像文件创建映像。有关更多信息，请参阅 EC2InstanceProfileForImageBuilder 策略。	2024 年 12 月 30 日

功能发布：ISO 到 AMI	Image Builder 现在可以根据微软 Windows 11 及更高版本的客户机操作系统的官方 ISO 磁盘文件创建映像。	2024 年 12 月 30 日
STIG 第四季度更新	更新了 Linux STIG 版本并应用了 2024 年第四季度发布的 STIGS。添加了有关 Linux 组件的两个新输入参数的信息。	2024 年 12 月 10 日
IAM 政策更新：实例配置文件策略	更新了实例配置文件策略以授予获取 AWS Marketplace 组件的访问权限。有关更多信息，请参阅 EC2InstanceProfile ForImageBuilder 策略。	2024 年 12 月 2 日
功能发布： AWS Marketplace 组件	增加了对 AWS Marketplace 软件组件的支持。	2024 年 12 月 2 日
功能发布：支持 macOS	增加了对 macOS 镜像的支持。	2024 年 10 月 22 日
文档中对逻辑运算符的支持 AWS Task Orchestrator and Executor	使用逻辑运算符在组件文档中添加或修改条件表达式。	2024 年 8 月 16 日
文档中对条件构造的支持 AWS Task Orchestrator and Executor	使用诸如“if”语句之类的条件构造来指导组件文档中组件操作的控制流程。	2024 年 8 月 16 日
文档对比较运算符的支持 AWS Task Orchestrator and Executor	使用比较运算符比较组件文档中的值。	2024 年 8 月 16 日
新增了 Assert 操作模块	在 `General execution` 下增加了 `ExecuteDocument` 操作模块的文档。	2024 年 8 月 14 日

操作系统支持	新增了对 RHEL 9 和 Ubuntu 24.04 LTS 的支持。	2024 年 8 月 2 日
文档更新：重新整理内容	重新整理了文档，以改进演示和导航。	2024 年 6 月 21 日
STIG 第二季度更新	更新了 Linux STIG 版本，并对 2024 年第二季度发行版应用了 STIG。新增了对 RHEL 9、CentOS Stream 9 和 Ubuntu 22.04 的支持。Windows 版本没有做任何变更。	2024 年 5 月 10 日
STIG 第一季度更新	更新了 Linux STIG 版本，并对 2024 年第一季度发行版应用了 STIG。Windows 版本没有做任何变更。	2024 年 2 月 23 日
STIG 第一季度更新	更新了 Linux STIG 版本，并对 2024 年第一季度发行版应用了 STIG。Windows 版本没有做任何变更。	2024 年 2 月 23 日
功能发布：映像工作流管理	借助映像工作流，您可以更灵活、更清晰和更好地控制映像创建过程。您可以自定义工作流的构建和测试步骤，也可以使用 Image Builder 默认工作流。	2023 年 12 月 12 日
STIG 第四季度更新	更新了 STIG 版本，并对 2023 年第四季度发行版应用了 STIG。Windows 版本没有做任何变更。还针对新组件、软件和基准测试编号更新了 Linux 和 Windows SCAP。	2023 年 12 月 7 日

功能发布：映像生命周期管理	借助映像生命周期管理策略和规则，您可以定义资源管理策略，以确保过时的映像及其关联资源完成标记和删除过程。	2023 年 11 月 17 日
IAM 策略更新：服务角色	针对实例放置支持更新了服务相关角色策略。有关更多信息，请参阅 AWSServiceRoleForImageBuilder 策略。	2023 年 10 月 19 日
STIG 第三季度更新	更新了 STIG 版本，并对 2023 年第三季度发行版应用了 STIG。此外还更新了消息传递，以澄清不会自动安装第三方软件包，只有极少数例外。所有跳过的内容都会 STIGs 被记录下来。	2023 年 10 月 5 日
新的 STIG 版本	更新了 STIG 版本，并对 2023 年第二季度发行版应用了 STIG。	2023 年 5 月 3 日
新的 STIG 版本	更新了 STIG 版本，并对 2023 年第一季度发行版应用了 STIG。添加了对的支持 AL2023。	2023 年 4 月 14 日
更新支持的区域 AWSTOE	增加了对以下地区的 AWSTOE 支持 AWS 区域：亚太地区（海得拉巴）、亚太地区（雅加达）、欧洲（苏黎世）、欧洲（西班牙）和中东（阿联酋）。	2023 年 4 月 13 日

AWSTOE 应用程序下载更新	更新了在 Windows 上下载 AWSTOE 安装的签名。还更新了 TLS，请注意，从 S3 存储桶下载应用程序现在需要使用 TLS 1.2 或更高版本。	2023 年 3 月 31 日
功能发布：增强版构建流程	在映像构建版本详情的新工作流程选项卡中添加了映像构建的运行时详情。改进了问题排查构建信息。	2023 年 3 月 30 日
功能发布：CVE 检测和报告	对于已激活 Amazon Inspector 扫描的账户，Image Builder 可以在新映像（包括存储在 Amazon ECR 中的容器映像）构建过程的测试阶段捕获来自 Amazon Inspector 的常见脆弱性和风险（CVE）调查发现。Image Builder 会创建调查发现的快照以支持详情分析。Image Builder 还会报告可按账户、管道或映像筛选的调查发现计数，并能够深入了解详情。	2023 年 3 月 30 日
添加了版本历史记录	在 Windows 和 Linux 部分中添加了版本历史记录。	2023 年 2 月 17 日
新的 STIG 版本	更新了 STIG 版本，并对 2022 年第四季度发行版应用了 STIG。	2023 年 2 月 1 日

功能发布： AWS Marketplace 集成和 CIS 强化	增加了 AWS Marketplace 集成，可轻松查找订阅的映像并将其用作新自定义映像的基准，包括 CIS 加固映像和互联网安全中心提供的新 CIS 加固组件。	2023 年 1 月 13 日
CIS 强化组件	增加了由 CIS 拥有和维护的 CIS 强化组件。	2023 年 1 月 13 日
新的 STIG 版本	引入了 Ubuntu 支持，更新了 STIG 版本，并对 2022 年第二季度发行版应用了 STIGS。	2022 年 7 月 20 日
文档更新：创建 YAML 组件文档页面导航	将 Create YAML 组件文档内容移至其自己的页面，并更新了其他页面以引用该内容。	2022 年 6 月 7 日
新的 STIG 版本	更新了 STIG 版本，并对 2022 年第一季度发行版应用了 STIG。	2022 年 4 月 25 日
添加了 ExecuteDocument 动作模块	在 `General execution` 下增加了 `ExecuteDocument` 操作模块的文档。	2022 年 3 月 28 日
功能发布：支持更快地启动 Windows AMI	添加了分发配置设置，以支持更快地启动 Windows AMIs。	2022 年 2 月 21 日
维护版本：更新 AWSTOE 二进制指纹	更新了 AWSTOE 签名者证书的二进制指纹。	2022 年 2 月 18 日
功能发布：配置输入 AWSTOE	增加了对使用 JSON 配置文件作为 AWSTOE run命令输入的支持。	2022 年 2 月 3 日

新的 STIG 版本	更新了 STIG 版本，并对 2021 年第四季度发行版应用了 STIG。还添加了有关新的 SCAP Compliance Checker (SCC) 组件的部分。	2021 年 12 月 22 日
功能发布：虚拟机 Import/Export (VMIE) 集成	增加了对通过所有渠道（控制台）导入虚拟机的支持API/CLI, etc.), and for VM export via API/CLI。Image Builder 控制台目前无法导出 VM。	2021 年 12 月 20 日
功能发布： AWS Organizations 和的 AMI 共享 OUs	更新了分发配置，增加了对 AMIs 与 AWS Organizations 和共享输出的支持 OUs。	2021 年 11 月 24 日
文档更新：更新组件期和组件阶段	扩展了 Image Builder 中组件阶段的内容，以及这些阶段与 AWSTOE 组件阶段的交互方式。	2021 年 9 月 22 日
文档更新：添加 CloudTrail 集成内容	增加了监控摘要和 CloudTrail 集成内容。	2021 年 9 月 17 日
新的 STIG 版本	更新了 STIG 版本，并对 2021 年第三季度发行版应用了 STIG。	2021 年 9 月 10 日
功能发布：Amazon EventBridge 集成	增加了 EventBridge 支持，使您可以将 Image Builder 与来自相关的事件连接起来 AWS 服务，并根据中定义的规则启动事件 EventBridge。	2021 年 8 月 18 日
文档更新：重新排序页面 AWSTOE	为了清晰起见，重新排列了 AWSTOE 页面。	2021 年 8 月 11 日

功能发布：参数化组件和其他实例配置	增加了对指定参数以自定义配方组件的支持。扩展了用于构建和测试映像的 EC2 实例的配置，包括可以指定在启动时运行的命令，以及更好地控制 Systems Manager 代理的安装和删除。	2021 年 7 月 7 日
新的 STIG 版本	更新了 STIG 版本，并对 2021 年第二季度发行版应用了 STIG。	2021 年 6 月 30 日
增强功能：标记增强功能	改进了围绕资源标记的消息。	2021 年 6 月 25 日
功能发布：启动模板集成	增加了对在“分配”设置中使用 Amazon EC2 启动模板进行 AMI 分配的支持。	2021 年 4 月 7 日
功能发布：容器构建增强功能	增加了对配置块储存设备映射和指定 AMIs 用作容器构建的基础映像的支持。	2021 年 4 月 7 日
新的 STIG 版本	更新了 STIG 版本并应用了 STIG。	2021 年 3 月 5 日
更新 cron 表达式	更新了 Image Builder cron 处理，可将 cron 表达式的粒度提高到分钟级别，并使用标准的 cron 计划引擎。使用新格式更新了示例。	2021 年 2 月 8 日

功能发布：容器支持	增加了对使用 Image Builder 创建 Docker 容器映像的支持，以及在 Amazon Elastic Container Registry (Amazon ECR) 上注册和存储生成的映像。对内容进行了重新编排，以反映新的功能并适应未来的功能增加。	2020 年 12 月 17 日
新编 cron 文档	本页面现在重点介绍有关 cron 如何处理 Image Builder 管道构建的更多信息，并包含有关 UTC 时间的详细信息。删除了不允许用于特定字段的通配符。示例现在包括控制台和 CLI 的表达式示例。	2020 年 11 月 13 日
控制台版本 2.0：更新了管道编辑功能	对管道入门和创建管道教程以及管理映像管道页面中的内容进行了更改，以纳入新的控制台功能和流程。	2020 年 11 月 13 日
新的 STIG 版本	更新了 STIG 版本并应用了 STIG。注意-列表格式已更改为默认应用 STIGs 的显示格式。	2020 年 10 月 15 日
Support 支持在中循环构造 AWSTOE	创建循环构造以定义 AWSTOE 应用程序中重复的指令序列。	2020 年 7 月 29 日
Support 支持 AWSTOE 组件的本地开发	使用 AWSTOE 应用程序在本地开发和测试映像组件。	2020 年 7 月 28 日
已加密 AMIs	EC2 Image Builder 增加了对加密 AMI 分配的支持。	2020 年 7 月 1 日
AutoScaling 弃用	不赞成使用 AutoScaling 来启动实例。	2020 年 6 月 15 日

Support 支持通过以下方式进行连接 AWS PrivateLink	您可以通过创建接口 VPC 端点在 VPC 和 EC2 Image Builder 之间建立私有连接。接口端点由一项技术提供支持 AWS PrivateLink，该技术使您 APIs 无需互联网网关、NAT 设备、VPN 连接或 Di AWS rect Connect 连接即可私密访问 Image Builder。您的 VPC 中的实例不需要公有 IP 地址即可与 Image Builder 通信 APIs。VPC 和 Image Builder 之间的流量不会脱离 Amazon 网络。	2020 年 6 月 10 日
新的 STIG 版本	更新了 STIG 版本并应用了 STIG。	2020 年 1 月 23 日
故障排查	增加了一般故障排除场景。	2020 年 1 月 22 日
STIG 组件	您可以使用 STIG 组件创建符合 STIG 标准的 AWSTOE 图像。	2020 年 1 月 22 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。