
AWS Systems Manager

用户指南



AWS Systems Manager: 用户指南

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

什么是 AWS Systems Manager ?	1
如何使用	1
功能	3
资源组	3
见解	3
操作	3
共享资源	4
入门	4
访问 Systems Manager	5
定价	5
我们希望听到您的意见和建议	5
相关内容	5
设置	6
先决条件	6
配置访问权限	9
任务 1：配置用户对 Systems Manager 的访问权限	9
任务 2：为 Systems Manager 创建实例配置文件	10
任务 3：创建使用 Systems Manager 实例配置文件的 Amazon EC2 实例	11
可选的访问权限配置	11
设置 Systems Manager 的 VPC 端点	13
创建 Systems Manager 的 VPC 端点	14
安装和配置 SSM 代理	14
在 Windows 实例上安装和配置 SSM 代理	15
在 Linux 实例上安装和配置 SSM 代理	17
订阅 SSM 代理通知	29
SSM 代理的最低 S3 存储桶权限	29
设置混合环境	31
为混合环境创建 IAM 服务角色	31
为混合环境创建托管实例激活	32
在 Windows 混合环境中的服务器和虚拟机上安装 SSM 代理	34
在 Linux 混合环境中的服务器和虚拟机上安装 SSM 代理	35
合作伙伴和产品集成	38
从 GitHub 和 Amazon S3 运行脚本	38
从 GitHub 运行脚本	38
从 Amazon S3 运行脚本	43
使用 Run Command 为 EBS 卷拍摄启用 VSS 的快照	47
如何使用	47
开始前的准备工作	47
创建启用 VSS 的 EBS 快照	50
通过使用 AWSEC2-ManagedVssIO SSM 文档 (高级) 创建启用 VSS 的 EBS 快照	54
从启用 VSS 的 EBS 快照还原卷	54
将 Chef InSpec 配置文件与 Systems Manager Compliance 结合使用	55
如何使用	56
运行 InSpec 合规性扫描	56
资源组	59
见解	60
清单	60
开始使用 Inventory	60
关于 Systems Manager Inventory	60
配置清单收集	69
配置清单的资源数据同步	71
查询清单收集	73
删除自定义清单	74
清单演练	82

排除 Systems Manager Inventory 故障	91
合规性	92
配置合规性入门	92
关于配置合规性	94
修复合规性问题	96
配置合规性演练	97
操作	100
自动化	100
AWS Systems Manager 自动化概念	100
Automation 使用案例	101
Automation 快速入门	103
设置 Automation	107
Automation 演练	114
使用 Automation 文档	118
Automation 示例	133
Automation 系统变量	152
排除 Systems Manager Automation 的故障	160
Run Command	171
设置 Run Command	172
运行命令	181
了解命令状态	189
Run Command 演练	191
对 Run Command 进行故障排除	201
Patch Manager	204
Patch Manager 支持的操作系统	204
如何使用	205
介绍如何修补实例的 SSM 文档概述	215
关于 SSM 文档 AWS-RunPatchBaseline	217
使用 Patch Manager	219
Patch Manager 演练	225
Patch Manager 的 AWS CLI 命令	234
Maintenance Window	246
Controlling Access	247
使用 Maintenance Window	255
Maintenance Window 演练	259
状态管理器	277
关于 状态管理器	278
示例 状态管理器 文档	279
使用关联	280
状态管理器 演练	285
共享资源	289
托管实例	289
激活	289
文档	289
Systems Manager 预定义文档	290
SSM 文档架构和功能	291
SSM 文档语法	292
创建 Systems Manager 文档	297
为文档添加标签	299
共享 Systems Manager 文档	302
创建复合文档	308
从远程位置运行文档	309
SSM 文档插件参考	312
自动化文档参考	334
Parameter Store	361
Systems Manager 参数入门	362
关于 Systems Manager 参数	362

使用 Systems Manager 参数	365
设置 Systems Manager 参数	377
Parameter Store 演练	382
监视实例	389
将日志发送到 CloudWatch Logs (SSM 代理)	389
将日志发送到 CloudWatch Logs (CloudWatch 代理)	390
将 Windows Server 实例日志收集迁移到 CloudWatch 代理	391
将 CloudWatch 代理配置设置存储到 Parameter Store 中	394
回滚到使用 SSM 代理进行日志收集	394
使用 AWS CloudTrail 记录 API 调用	395
CloudTrail 中的 Systems Manager 信息	395
了解 Systems Manager 日志文件条目	396
身份验证和访问控制	398
身份验证	398
访问控制	399
访问管理概述	399
资源和操作	400
了解资源所有权	402
管理对资源的访问	402
指定策略元素：资源、操作、效果和委托人	403
在策略中指定条件	404
使用基于身份的策略 (IAM 策略)	404
使用 AWS Systems Manager 控制台所需的权限	405
适用于 AWS Systems Manager 的 AWS 托管 (预定义) 策略	405
客户托管策略示例	406
使用服务相关角色	407
用于 Systems Manager 的服务相关角色权限	407
为 Systems Manager 创建服务相关角色	408
编辑用于 Systems Manager 的服务相关角色	408
删除用于 Systems Manager 的服务相关角色	408
AWS Systems Manager 权限参考	408
AWS Systems Manager 参考	418
Cron 和 Rate 表达式	418
适用于关联的 Cron 和 Rate 表达式	419
适用于Maintenance Window的 Cron 和 Rate 表达式	420
有关 Cron 和 Rate 表达式的一般信息	420
Ec2messages 和其他 API 调用	422
使用案例和最佳实践	424
文档历史记录	426
早期更新	427
AWS Glossary	437

什么是 AWS Systems Manager ?

AWS Systems Manager 是一个功能集合，用于大规模配置和管理您的 Amazon EC2 实例、本地服务器与虚拟机及其他 AWS 资源。Systems Manager 包括一个统一接口，您可通过该接口轻松地将操作数据集中到一起，并跨 AWS 资源自动完成任务。Systems Manager 缩短了检测和解决基础设施中存在的操作问题所需的时间。Systems Manager 让您能够全面了解您的基础设施性能和配置，简化资源和应用程序管理，并简化基础设施的大规模操作和管理。

Note

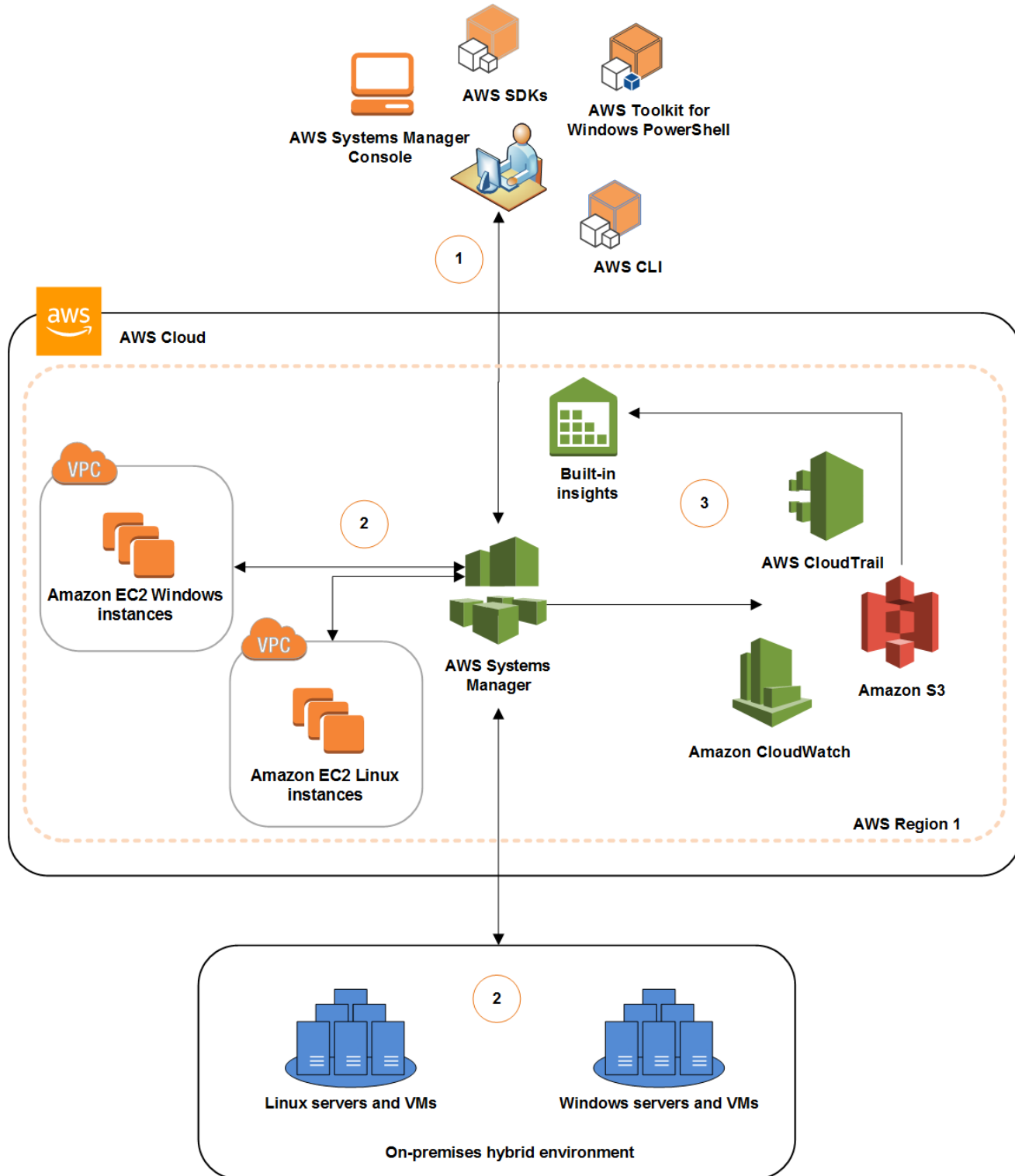
AWS Systems Manager 以前称为“Amazon EC2 Systems Manager”和“Amazon Simple Systems Manager”。

如何使用

图 1 显示了 Systems Manager 在执行操作（如向您的服务器队列发送命令）或执行在本地服务器上运行的应用程序清单时所应用的其他流程的一般示例。示例 Run Command 或 Maintenance Windows 中的每种 Systems Manager 功能使用类似设置、执行、处理和报告流程。

1. 配置 Systems Manager：使用 Systems Manager 控制台、SDK、AWS CLI 或 AWS Toolkit for Windows PowerShell，配置、计划、自动执行和执行您要针对 AWS 资源所执行的操作。
2. 验证和处理：Systems Manager 会验证包括权限在内的配置，并向在您混合环境的实例或服务器上运行的 SSM 代理发送请求。SSM 代理执行指定的配置更改。
3. 报告：SSM 代理会报告对 AWS 云中的 Systems Manager 所做的配置更改和所执行操作的状态。然后，Systems Manager 会向用户和各种 AWS 服务（如果已配置）发送该状态。

图 1：Systems Manager 流程的一般示例



功能

Systems Manager 包括以下功能：

主题

- [资源组 \(p. 3\)](#)
- [见解 \(p. 3\)](#)
- [操作 \(p. 3\)](#)
- [共享资源 \(p. 4\)](#)

资源组

AWS 资源组：AWS 资源是您可以在 AWS 中使用的一种实体，如 Amazon Elastic Compute Cloud (Amazon EC2) 实例、Amazon Elastic Block Store (Amazon EBS) 卷、安全组或 Amazon Virtual Private Cloud (VPC)。资源组是同一 AWS 区域中与查询中提供的条件相匹配的 AWS 资源的集合。您可以在资源组控制台构建查询，或者将查询作为自变量传递给 AWS CLI 中的资源组命令。借助资源组，您可以创建自定义控制台，根据在标签中指定的条件来组织和整合信息。您也可以将组作为在 AWS Systems Manager 中查看监控和配置见解的基础。

见解

Systems Manager 提供以下功能，用于集中查看有关 AWS 资源的数据。

内置见解

[见解](#)可显示关于 AWS 资源组中资源的详细信息，如 AWS CloudTrail 日志、依据 AWS Config 规则的评估结果以及 AWS Trusted Advisor 报告。见解一次显示关于一个选定资源组的信息。

CloudWatch 控制面板

[Amazon CloudWatch 控制面板](#)是 CloudWatch 控制台中的可自定义主页，可用于在一个视图中监控您的资源，即便是那些分布到不同区域的资源，也能对其进行监控。您可以使用 CloudWatch 控制面板创建 AWS 资源的指标和警报的自定义视图。

Inventory 管理

[Inventory Manager](#) 可自动执行从托管实例中收集软件清单的流程。(p. 60)您可以使用 Inventory Manager 在托管实例上收集有关应用程序、文件、组件、修补程序等对象的元数据。

配置合规性

使用 [Systems Manager 配置合规性 \(p. 92\)](#)扫描托管实例队列，了解补丁合规性和配置不一致性。您可以从多个 AWS 账户和区域中收集并聚合数据，然后深入了解不合规的特定资源。默认情况下，配置合规性将显示有关 Patch Manager 修补和 状态管理器 关联的合规性数据。您也可以根据 IT 或业务要求自定义服务并创建自己的合规性类型。

操作

Systems Manager 提供以下可对您的 AWS 资源执行操作的功能。

自动化

使用 [Systems Manager 自动化 \(p. 100\)](#)可自动执行常见的维护和部署任务。您可以使用自动化创建和更新 Amazon 系统映像、应用驱动程序和代理的更新、在 Windows 实例上重置密码、在 Linux 实例上重置 SSH 密钥，并应用操作系统补丁或应用程序更新。

Run Command

使用 [Systems Manager Run Command \(p. 171\)](#) 以远程方式安全且大规模地管理托管实例的配置。使用 Run Command 在几十个或数百个实例的目标集中执行按需更改，例如更新应用程序或运行 Linux Shell 脚本和 Windows PowerShell 命令。

补丁管理

使用 [Patch Manager \(p. 204\)](#) 可自动执行修补托管实例的过程。利用此功能，您可以扫描实例中是否有缺失的补丁，然后单独应用缺失的补丁或使用 Amazon EC2 实例标签将这些补丁应用于大型实例组。对于安全性补丁，Patch Manager 使用补丁基准，该基准包含用于在补丁发布几天内自动批准补丁的规则以及一系列已批准和已拒绝的补丁。从为实例配置的默认补丁存储库安装安全性补丁。您可以通过安排修补作为 Systems Manager Maintenance Window 任务运行来定期安装安全性补丁。对于 Linux 操作系统，您可以定义的存储库应该作为补丁基准的一部分用于修补操作。这样，您可以确保更新仅从信任的存储库安装，而与在实例上配置的存储库无关。对于 Linux，您还能够更新实例上的任何包，而不仅仅是被归类为操作系统安全更新的包。

Maintenance Window

使用 [Maintenance Window \(p. 246\)](#) 可以设置托管实例的周期性计划，以便执行诸如安装补丁和更新等管理任务，而不会中断业务关键性操作。

状态管理

使用 [Systems Manager 状态管理器 \(p. 277\)](#) 可自动使您的托管实例保持在一个预先设定的状态。您可以使用 状态管理器 确保您的实例在启动时使用特定软件进行引导，加入某个 Windows 域 (仅限 Windows 实例) 或使用特定的更新版本修补软件。

共享资源

Systems Manager 使用以下共享资源来管理和配置您的 AWS 资源。

托管实例

[托管实例 \(p. 6\)](#) 是任何 Amazon EC2 实例，或针对 Systems Manager 配置的混合环境中的本地计算机 (服务器或虚拟机 [VM])。要设置托管实例，您需要在计算机上安装 SSM 代理 (如果默认情况下未安装) 并配置 AWS Identity and Access Management (IAM) 权限。本地计算机也需要一个激活代码。

激活

要在混合环境中将服务器和虚拟机设置为托管实例，您需要创建托管实例 [激活 \(p. 31\)](#)。完成激活后，您将收到一个激活代码和 ID。此代码/ID 组合具有 Amazon EC2 访问 ID 和私有密钥的功能，可提供从托管实例对 Systems Manager 服务的安全访问。

Systems Manager 文档

[Systems Manager 文档 \(p. 289\)](#) (SSM 文档) 定义了 Systems Manager 执行的操作。SSM 文档可以是 状态管理器 和 Run Command 使用的命令文档，也可以是 Systems Manager 自动化使用的自动化文档。Systems Manager 包括几十个预先配置的文档，您可以通过在运行时指定参数进行使用。文档可以采用 JSON 或 YAML 表示，并包括您指定的步骤和参数。

Parameter Store

[Parameter Store \(p. 361\)](#) 提供安全的分层存储，用于配置数据管理和密钥管理。也可以将密码、数据库字符串和许可证代码等数据存储为参数值。可以将值存储为纯文本或加密数据。然后，可以使用创建参数时指定的唯一名称来引用对应值。

入门

要开始使用 Systems Manager，请执行以下操作：

- 确保您已满足 Systems Manager 先决条件
- 配置角色和权限
- 在实例上安装 SSM 代理 (如有必要)

如果您想使用 Systems Manager 管理本地服务器和虚拟机，则还必须创建托管实例激活。

[设置 AWS Systems Manager \(p. 6\)](#) 中介绍了这些任务。

访问 Systems Manager

您可以使用以下任意界面访问 Systems Manager：

- [AWS Systems Manager 控制台](#) — 提供可用于访问 Systems Manager 的 Web 界面。
- AWS 命令行界面 (AWS CLI) - 提供了用于众多 AWS 服务 (包括 Systems Manager) 的命令，并且在 Windows、Mac 和 Linux 上受支持。有关更多信息，请参阅 [AWS 命令行界面](#)。
- AWS Tools for Windows PowerShell — 为众多 AWS 服务 (包括 Systems Manager) 提供命令。相关详情，请参阅 [AWS Tools for Windows PowerShell](#)。
- AWS 软件开发工具包 - 提供了特定于语言的 API，并关注许多连接详细信息，例如计算签名、处理请求重试和错误处理。有关更多信息，请参阅 [AWS 软件开发工具包](#)。

定价

提供的 Systems Manager 功能和共享组件无需额外费用。您仅需为实际使用的 AWS 资源付费。

我们希望听到您的意见和建议

我们欢迎您提供反馈。要与我们联系，请访问 [AWS Systems Manager 论坛](#)。

相关内容

以下参考文档中也介绍了 Systems Manager。

- [Amazon EC2 Systems Manager API Reference](#)
- [Systems Manager AWS Tools for Windows PowerShell](#)
- [Systems Manager AWS CLI 参考](#)
- [AWS 软件开发工具包](#)
- [AWS Systems Manager 限制](#)

设置 AWS Systems Manager

本部分介绍设置 AWS Systems Manager 的任务和先决条件。使用下表帮助您快速开始。

您要使用 Systems Manager 做什么？	设置任务
测试/使用	<ol style="list-style-type: none">1. 验证权限并创建实例配置文件角色 (p. 9)。2. 根据最新 Amazon Linux、Ubuntu Server 18.04 LTS 或 Windows AMI 创建一些 Amazon EC2 测试实例 (免费套餐)。3. 测试 Systems Manager。以下演示可帮助您快速入门。 Note 其中部分演练需要额外的设置任务才能完成，如增加权限等。<ul style="list-style-type: none">• Run Command (EC2 控制台) : Linux 或 Windows• Run Command 演练 (p. 191) (AWS CLI 或 AWS Tools for Windows PowerShell)• 状态管理器 演练 (p. 285)• Parameter Store 演练 (p. 382)• Inventory Manager 演练 (p. 82)• Automation 演练 (p. 114)• Maintenance Window 演练 (p. 259)• Patch Manager 演练 (p. 225)
使用 Systems Manager 管理和配置现有的 EC2 实例	<ol style="list-style-type: none">1. 验证权限并创建实例配置文件角色 (p. 9)。2. 验证您的 EC2 实例是否符合 Systems Manager 要求 (p. 6)。3. (仅限 Linux) 安装 SSM 代理 (p. 9)。
使用 Systems Manager 在混合环境中管理和配置服务器和 VM	<ol style="list-style-type: none">1. 验证权限并创建实例配置文件角色 (p. 9)。2. 验证混合环境中的服务器和 VM 是否符合 Systems Manager 要求 (p. 6)。3. 执行混合环境中的托管实例的设置和激活任务 (p. 31)。

Systems Manager 先决条件

Systems Manager 包括下列先决条件。

要求	描述
支持的操作系统 (Windows)	实例必须运行支持的 Windows Server 版本：Windows Server 2003 至 Windows Server 2016，包括 R2 版本。

要求	描述
	<p>Note</p> <p>Patch Manager 功能只支持其中部分版本的 Windows Server 操作系统。有关信息，请参阅 Patch Manager 支持的操作系统 (p. 204)。</p>
支持的操作系统 (Linux)	<p>实例必须运行支持的 Linux 版本。</p> <p>Note</p> <p>Patch Manager 功能当前不支持所有以下 Linux 操作系统。有关信息，请参阅 Patch Manager 支持的操作系统 (p. 204)。</p> <p>64 位和 32 位系统</p> <ul style="list-style-type: none">• Amazon Linux 基本 AMI 2014.09、2014.03 或更高版本• Ubuntu Server 18.04 LTS、16.04 LTS、14.04 LTS 或 12.04 LTS• Red Hat Enterprise Linux (RHEL) 6.5• CentOS 6.3 或更高版本 <p>仅 32 位系统</p> <ul style="list-style-type: none">• Raspbian Jessie• Raspbian Stretch <p>仅 64 位系统</p> <ul style="list-style-type: none">• Amazon Linux 2015.09、2015.03 或更高版本• Amazon Linux 2• Red Hat Enterprise Linux (RHEL) 6.0、7.4 或 7.5• CentOS 7.1 或更高版本• SUSE Linux Enterprise Server (SLES) 12 或更高版本
支持的区域	<p>Systems Manager 在这些区域中可用。</p> <p>对于混合环境中的服务器和虚拟机，我们建议您选择与您的数据中心或计算环境最接近的区域。</p>

要求	描述
访问 Systems Manager	<p>Systems Manager 需要一个针对将处理命令的实例的 IAM 角色和一个针对执行命令的用户的独立角色。两个角色都需要权限策略才能通过 Systems Manager API 进行通信。您可以选择使用 Systems Manager 托管策略或创建自己的角色并指定权限。有关更多信息，请参阅 配置对 Systems Manager 的访问权限 (p. 9)。</p> <p>如果您要配置本地服务器，或要使用 Systems Manager 配置虚拟机，那么还必须配置 IAM 服务角色。有关更多信息，请参阅 为混合环境创建 IAM 服务角色 (p. 31)。</p>
SSM 代理 (EC2 Windows 实例)	<p>SSM 代理负责处理 Systems Manager 请求并按照请求中指定的方式配置您的计算机。默认情况下，SSM 代理安装在 Windows Server 2016 实例以及通过 2016 年 11 月或之后发布的 Windows Server 2003-2012 R2 AMI 创建的实例上。</p> <p>2016 年 11 月之前发布的 Windows AMI 使用 EC2Config 服务处理请求并配置实例。</p> <p>除非您出于特定原因需要使用 EC2Config 服务或早期版本的 SSM 代理来处理 Systems Manager 请求，否则建议您下载最新版本的 SSM 代理并将其安装到每个 Amazon EC2 实例或托管实例 (混合环境中的服务器和 VM)。有关更多信息，请参阅 在 Windows 实例上安装和配置 SSM 代理 (p. 15)。</p>
SSM 代理 (EC2 Linux 实例)	<p>SSM 代理负责处理 Systems Manager 请求并按照请求中指定的方式配置您的计算机。默认情况下，Amazon Linux、Amazon Linux 2、Ubuntu Server 16.04 和 Ubuntu Server 18.04 LTS 基本 AMI 上安装了 SSM 代理。您必须在其他版本的 EC2 Linux 上手动安装 SSM 代理，包括 Amazon ECS 优化的 AMI 等非基本映像。有关更多信息，请参阅 在 Linux 实例上安装和配置 SSM 代理 (p. 17)。</p> <p>在 GitHub 上可以找到 SSM 代理的源代码，您可以调整代理以满足您的需求。我们鼓励您针对要包含的更改提交提取请求。但是，Amazon Web Services 当前不支持运行此软件的修改后副本。</p>
SSM 代理 (混合环境)	<p>混合环境中托管实例的 SSM 代理下载和安装过程与 Amazon EC2 实例的不同。有关更多信息，请参阅 在 Windows 混合环境中的服务器和虚拟机上安装 SSM 代理 (p. 34)。</p>
Windows PowerShell 3.0 或更高版本	<p>SSM 代理需要 Windows PowerShell 3.0 或更高版本才能在 Windows 实例上执行特定 SSM 文档 (例如，AWS-ApplyPatchBaseline 文档)。验证您的 Windows 实例是否在 Windows 管理框架 3.0 或更高版本上运行。该框架包括 PowerShell。有关更多信息，请参阅 Windows 管理框架 3.0。</p>

要求	描述
Internet 访问	验证 EC2 实例是否具有出站 Internet 访问权。无需入站 Internet 访问权。
配置监控和通知 (可选)	您可以配置 Amazon CloudWatch Events 来记录使用 Systems Manager 发送的命令的状态执行更改。您还可以配置 Amazon Simple Notification Service (Amazon SNS) 来发送有关特定命令状态更改的通知。有关更多信息，请参阅 了解命令状态 (p. 189) 。
Amazon S3 存储桶 (可选)	您可以在 Amazon Simple Storage Service (Amazon S3) 存储桶中存储 Systems Manager 输出。Amazon EC2 控制台中的输出将在 2500 个字符后被截断。此外，您可能希望创建 Amazon S3 键前缀 (子文件夹) 来帮助您整理输出。有关更多信息，请参阅 创建存储桶 。

有关 Systems Manager 限制的信息，请参阅 [AWS Systems Manager 限制](#)。要增加限制，请转到 [AWS 支持中心](#) 并提交限制增加请求表单。

配置对 Systems Manager 的访问权限

完成以下任务以配置 AWS Systems Manager 的访问权限。

Note

有关 Systems Manager 访问权限的更多信息，请参阅 [AWS Systems Manager 的身份验证和访问控制 \(p. 398\)](#)。

内容

- [任务 1：配置用户对 Systems Manager 的访问权限 \(p. 9\)](#)
- [任务 2：为 Systems Manager 创建实例配置文件 \(p. 10\)](#)
- [任务 3：创建使用 Systems Manager 实例配置文件的 Amazon EC2 实例 \(p. 11\)](#)
- [可选的访问权限配置 \(p. 11\)](#)

任务 1：配置用户对 Systems Manager 的访问权限

如果已为您的 IAM 用户账户、组或角色分配管理员权限，则您可以访问 Systems Manager。您可以跳过此任务。如果您没有管理员权限，则管理员必须将您的 IAM 用户账户、组或角色更新为包含以下权限：

- 访问资源组：您必须向您的 IAM 用户账户、组或角色添加 `resource-groups:*` 权限实体。有关更多信息，请参阅 [AWS 资源组用户指南中的设置权限](#)。
- 访问见解：您必须向您的用户账户、组或角色添加以下托管策略：
 - `AWSHealthFullAccess`
 - `AWSConfigUserAccess`
 - `CloudWatchReadOnlyAccess`

Note

以下策略涵盖清单访问权限和合规性。

- 访问操作和共享资源：您必须添加 AmazonSSMFullAccess 策略或 AmazonSSMReadOnlyAccess 策略。

有关如何更改 IAM 用户账户、组或角色的权限的信息，请参阅 IAM User Guide 中的 [更改 IAM 用户的权限](#)。

任务 2：为 Systems Manager 创建实例配置文件

默认情况下，Systems Manager 没有在其实例上执行操作的权限。您必须通过使用 IAM 实例配置文件来授予访问权限。实例配置文件是一个容器，可在启动时将 IAM 角色信息传递给 Amazon EC2 实例。

Note

如果要在混合环境中为 Systems Manager 配置服务器或虚拟机 (VM)，则无需为它们创建实例配置文件。相反，您必须将服务器和虚拟机配置为使用 IAM 服务角色。有关更多信息，请参阅 [为混合环境创建 IAM 服务角色](#) (p. 31)。

您可以为 Systems Manager 创建实例配置文件，方法是将定义所需权限的 AWS 托管策略附加到新角色或您已创建的角色。

在创建实例配置文件后，将它附加到要对其使用 Systems Manager 的实例。要将实例配置文件附加到现有实例，请参阅 [使用实例配置文件](#)。要在创建新实例时向它们附加实例配置文件，请参阅下一个主题 [任务 3：创建使用 Systems Manager 实例配置文件的 Amazon EC2 实例](#) (p. 11)。

Note

如果您更改了 IAM 实例配置文件，则可能需要花一些时间来更新实例凭证。在更新完成之前，SSM 代理将无法处理请求。要加快更新过程，您可以重新启动 SSM 代理或重新启动实例。

根据您是实例配置文件创建新角色还是将所需权限添加到现有角色，使用以下过程之一。

为 Systems Manager 托管实例创建实例配置文件

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择 Roles，然后选择 Create Role。
3. 在 Select type of trusted entity 页面上的 AWS Service 下，选择 EC2。

Note

如果显示了选择您的使用案例部分，请依次选择 EC2 Role for Simple Systems Manager (简单 Systems Manager 的 EC2 角色) 和下一步：权限。

4. 在 Attached permissions policy 页面上，确认是否已列出 AmazonEC2RoleforSSM，然后选择 Next: Review。
5. 在 Review 页面上，在 Role name 框中键入名称，然后键入描述。

Note

记下角色名称。在创建希望使用 Systems Manager 进行管理的新实例时，将选择该角色。

6. 选择 Create role。系统将让您返回到 Roles 页。

将 Systems Manager 托管实例的实例配置文件权限添加到现有角色

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择角色，然后选择要与 Systems Manager 操作的实例配置文件关联的现有角色。
3. 在权限选项卡上，选择附加策略。
4. 在附加策略页上，选中 AmazonEC2RoleforSSM 旁边的复选框，然后选择附加策略。

有关如何更新角色以包含可信实体或进一步限制访问的信息，请参阅[修改角色](#)。

有关如何将刚创建的角色附加到新实例或现有实例的信息，请参阅下一个主题[任务 3：创建使用 Systems Manager 实例配置文件的 Amazon EC2 实例](#) (p. 11)。

任务 3：创建使用 Systems Manager 实例配置文件的 Amazon EC2 实例

此过程介绍如何启动使用您在上一个主题[任务 2：为 Systems Manager 创建实例配置文件](#) (p. 10) 中创建的实例配置文件的 Amazon EC2 实例。您还可以将实例配置文件附加到现有实例。有关更多信息，请参阅 Amazon EC2 用户指南 中的[将 IAM 角色附加到实例](#)。

创建使用 Systems Manager 实例配置文件的实例

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. 在屏幕顶部的导航栏中，会显示当前区域。选择实例的[区域](#)。
3. 选择 Launch Instance。
4. 在选择一个 Amazon 系统映像 (AMI) 页面上，找到您要创建的实例类型的 AMI，然后选择选择。
5. 选择 Next: Configure Instance Details。
6. 在配置实例详细信息页面上的 IAM 角色下拉列表中，选择您在[任务 2：为 Systems Manager 创建实例配置文件](#) (p. 10) 的步骤中创建的实例配置文件。
7. 完成向导。

有关更多信息，请根据您的实例类型参阅下列主题之一：

- Linux：Amazon EC2 User Guide for Linux Instances 中的[使用启动实例向导启动实例](#)
- Windows：Amazon EC2 User Guide for Windows Instances 中的[使用启动实例向导启动实例](#)

如果您创建要使用 Systems Manager 配置的其他实例，则必须为每个实例指定实例配置文件。

可选的访问权限配置

通过执行本部分中的任务 1，您可以通过选择预先存在或托管的 IAM 用户策略来为用户授予访问权限。如果需要限制用户对 Systems Manager 和 SSM 文档的访问权限，您可以如本部分所述创建您自己的限制性用户策略。有关如何创建自定义策略的更多信息，请参阅[创建新策略](#)。

以下示例 IAM 策略允许用户执行以下操作。

- 列出 Systems Manager 文档和文档版本。
- 查看有关文档的详细信息。
- 使用策略中指定的文档发送命令。

文档名称由此条目确定：

```
arn:aws:ssm:us-east-2:::document/name_of_restrictive_document
```

- 发送命令到三个实例。

实例由第二个 Resource 部分中的以下条目确定：

```
"arn:aws:ec2:us-east-2:::instance/i-1234567890abcdef0",  
"arn:aws:ec2:us-east-2:::instance/i-0598c7d356eba48d7",  
"arn:aws:ec2:us-east-2:::instance/i-345678abcdef12345",
```


- 发送命令后查看有关命令的详细信息。
- 启动和停止 Automation 执行。
- 获取有关 Automation 执行的信息。

如果您要授予用户权限以使用本文档向该用户目前有权访问 (取决于其 AWS 用户账户) 的任何实例发送命令，您可以在 Resource 部分指定以下条目并删除其他实例条目。

```
"arn:aws:ec2:us-east-2:*:instance/*"
```

请注意，Resource 部分包含一个 Amazon S3 ARN 条目：

```
arn:aws:s3:::bucket_name
```

您还可为此条目设置格式，如下所示：

```
arn:aws:s3:::bucket_name/*

-or-

arn:aws:s3:::bucket_name/key_prefix_name
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ssm:ListDocuments",
        "ssm:ListDocumentsVersions",
        "ssm:DescribeDocument",
        "ssm:GetDocument",
        "ssm:DescribeInstanceInformation",
        "ssm:DescribeDocumentParameters",
        "ssm:DescribeInstanceProperties"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "ssm:SendCommand",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:ec2:us-east-2:*:instance/i-1234567890abcdef0",
        "arn:aws:ec2:us-east-2:*:instance/i-0598c7d356eba48d7",
        "arn:aws:ec2:us-east-2:*:instance/i-345678abcdef12345",
        "arn:aws:s3:::bucket_name",
        "arn:aws:ssm:us-east-2:*:document/name_of_restrictive_document"
      ]
    },
    {
      "Action": [
        "ssm:CancelCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "ec2:DescribeInstanceStatus",
      "Effect": "Allow",

```

```
    "Resource": "*"
  },
  {
    "Action": "ssm:StartAutomationExecution",
    "Effect": "Allow",
    "Resource": [
      "arn:aws:ssm::automation-definition/"
    ]
  },
  {
    "Action": "ssm:DescribeAutomationExecutions ",
    "Effect": "Allow",
    "Resource": [
      "*"
    ]
  },
  {
    "Action": [
      "ssm:StopAutomationExecution",
      "ssm:GetAutomationExecution"
    ],
    "Effect": "Allow",
    "Resource": [
      "arn:aws:ssm::automation-execution/"
    ]
  }
]
```

设置 Systems Manager 的 VPC 端点

您可以通过配置 AWS Systems Manager 以使用接口 VPC 端点来改善托管实例 (包括混合环境中的托管实例) 的安全状况。接口终端节点由 PrivateLink 提供技术支持, 该技术让您可以通过使用私有 IP 地址私下访问 Amazon EC2 和 Systems Manager API。PrivateLink 将托管实例、Systems Manager 和 EC2 之间的所有网络流量限制在 Amazon 网络内 (托管实例无法访问 Internet)。而且, 您无需 Internet 网关、NAT 设备或虚拟专用网关。

不要求您配置 PrivateLink, 但推荐进行配置。有关 PrivateLink 和 VPC 端点的更多信息, 请参阅[通过 PrivateLink 访问 AWS 服务](#)。

开始前的准备工作

在配置 Systems Manager 的 VPC 端点之前, 请注意以下限制。

- VPC 终端节点不支持 Active Directory 目录服务。如果您配置托管实例以使用 VPC 终端节点, 则无法使用此服务。

Note

本用户指南可能没有反映 Active Directory 目录服务中的最新开发成果。为此, 我们鼓励您参阅 [Active Directory 目录服务](#) 用户指南以获得 VPC 终端节点支持。

- VPC 端点当前不支持跨区域请求, 从而确保在您的存储桶所在区域内创建终端节点。您可以使用 Amazon S3 控制台或 [get-bucket-location](#) 命令来查找存储桶的位置。使用区域特定的 Amazon S3 终端节点访问存储桶; 例如, `mybucket.s3-us-west-2.amazonaws.com`。有关 Amazon S3 的区域特定的终端节点的更多信息, 请参阅 Amazon Web Services General Reference 中的 [Amazon Simple Storage Service \(S3\)](#)。如果您使用 AWS CLI 向 Amazon S3 发起请求, 请将默认区域设置为您的存储桶所在的区域, 或在请求中使用 `--region` 参数。
- VPC 端点仅通过 Route 53 支持 Amazon 提供的 DNS。如果您希望使用自己的 DNS, 可以使用条件 DNS 转发。有关更多信息, 请参阅 Amazon VPC User Guide 中的 [DHCP 选项集](#)。

- 附加到 VPC 端点的安全组必须允许从托管实例的私有子网通过端口 443 进行传入连接。如果不允许传入连接，则托管实例无法连接到 SSM 和 EC2 端点。

创建 Systems Manager 的 VPC 端点

使用以下过程为创建三个独立的 Systems Manager 的 VPC 端点。Systems Manager 需要这三个终端节点都能 VPC 中运行。此步骤可链接到 Amazon VPC User Guide 中的相关步骤。

创建 Systems Manager 的 VPC 端点

1. 使用 [创建接口终端节点](#) 来创建以下终端节点：

- `com.amazonaws.region.ssm`：Systems Manager 服务的终端节点。
- `com.amazonaws.region.ec2messages`：Systems Manager 使用此终端节点从 SSM 代理调用 Systems Manager 服务。
- `com.amazonaws.region.ec2`：如果您使用 Systems Manager 创建启用 VSS 的快照，则需要确保您具有连接到 EC2 服务的终端节点。未定义 EC2 终端节点时，枚举所附加 EBS 卷的调用将失败，这会导致 Systems Manager 命令失败。有关使用 Systems Manager 创建启用 VSS 的快照的信息，请参阅 [使用 Run Command 为 EBS 卷拍摄启用 VSS 的快照 \(p. 47\)](#)。

##代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 US East (Ohio) Region。有关受支持 **##** 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

2. 使用 [创建网关终端节点](#) 来创建 Amazon S3 的终端节点。Systems Manager 使用此终端节点上传 Amazon S3 输出日志并更新 SSM 代理。

安装和配置 SSM 代理

AWS Systems Manager 代理 (SSM 代理) 是在您的 Amazon EC2 实例和为 Systems Manager 配置的混合实例 (以下简称混合实例) 上运行的 Amazon 软件。SSM 代理在云中从 Systems Manager 服务处理请求并按照请求中指定的方式配置计算机。SSM 代理使用 EC2 消息收发服务将状态和执行信息发送回 Systems Manager 服务。如果您对流量进行监控，就会发现实例会与 `ec2messages.*` 终端节点进行通信。有关更多信息，请参阅 [参考：Ec2messages 和其他 API 调用 \(p. 422\)](#)。

默认情况下，SSM 代理安装在以下 Amazon EC2 Amazon Machine Image (AMI) 上：

- Windows Server (所有 SKU)
- Amazon Linux
- Amazon Linux 2
- Ubuntu Server 16.04
- Ubuntu Server 18.04

您必须手动将代理安装到其他 Linux AMI 创建的 Amazon EC2 实例上或本地环境中的 Linux 服务器或虚拟机上。

Note

混合实例的 SSM 代理下载和安装过程与 Amazon EC2 实例的不同。有关更多信息，请参阅 [在 Windows 混合环境中的服务器和虚拟机上安装 SSM 代理 \(p. 34\)](#)。

有关将 SSM 代理日志移植到 Amazon CloudWatch Logs 的信息，请参阅 [使用 AWS Systems Manager 监控实例 \(p. 389\)](#)。

使用以下过程可安装、配置或卸载 SSM 代理。本部分还包含有关配置 SSM 代理使用代理的信息。

内容

- [在 Windows 实例上安装和配置 SSM 代理 \(p. 15\)](#)
- [在 Linux 实例上安装和配置 SSM 代理 \(p. 17\)](#)
- [订阅 SSM 代理通知 \(p. 29\)](#)
- [SSM 代理的最低 S3 存储桶权限 \(p. 29\)](#)

在 Windows 实例上安装和配置 SSM 代理

默认情况下，SSM 代理安装在 Windows Server 2016 实例以及通过 2016 年 11 月或之后发布的 Windows Server 2003-2012 R2 AMI 创建的实例上。

2016 年 11 月之前发布的 Windows AMI 使用 EC2Config 服务处理请求并配置实例。

除非您出于特定原因需要使用 EC2Config 服务或早期版本的 SSM 代理来处理 Systems Manager 请求，否则建议您下载最新版本的 SSM 代理并将其安装到每个 Amazon EC2 实例或为 Systems Manager 配置的混合实例。

如果您需要更新 SSM 代理，我们建议您使用 状态管理器 在您的实例上自动更新可用的 SSM 代理 新版本。有关更多信息，请参阅 [演练：自动更新 SSM 代理 \(CLI\) \(p. 286\)](#)。

要查看有关不同版本 SSM 代理的详细信息，请参阅 [发行说明](#)。

主题

- [在 Windows 实例上安装和配置 SSM 代理 \(p. 15\)](#)
- [在 Windows 实例上查看 SSM 代理日志 \(p. 16\)](#)
- [配置 SSM 代理以使用适用于 Windows 实例的代理 \(p. 16\)](#)

在 Windows 实例上安装和配置 SSM 代理

默认情况下，SSM 代理 安装在 Windows Server 2016 实例上。默认情况下也安装在通过 2016 年 11 月或之后发布的 Windows Server 2003-2012 R2 AMI 创建的实例上。您无需在这些实例上安装 SSM 代理。如果您需要更新 SSM 代理，我们建议您使用 状态管理器 在您的实例上自动更新可用的 SSM 代理 新版本。有关更多信息，请参阅 [演练：自动更新 SSM 代理 \(CLI\) \(p. 286\)](#)。

如果您的实例是在 2016 年 11 月之前创建的 Windows Server 2003-2012 R2 实例，则 EC2Config 在您的实例上处理 Systems Manager 请求。我们建议您升级现有实例以使用最新版本的 EC2Config。您可以使用最新的 EC2Config 安装程序，在安装 EC2Config 的同时安装 SSM 代理。此并行版本 SSM 代理 与通过之前的 Windows AMI 创建的实例兼容，而且借助此代理，您能够使用 2016 年 11 月之后发布的 SSM 功能。有关如何安装最新版本的 EC2Config 服务的信息，请参阅 Amazon EC2 User Guide for Windows Instances 中的 [安装最新版的 EC2Config](#)。

如果需要，您也可以使用以下过程手动下载并安装最新版本的 SSM 代理。

手动下载并安装最新版本的 SSM 代理

1. 使用远程桌面或 Windows PowerShell 等工具登录实例。
2. 将最新版本的 SSM 代理下载到实例：

https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe

使用此 URL 可从任意 AWS 区域下载 SSM 代理。如果您要从特定区域下载代理，请改为使用特定于区域的 URL：

```
https://amazon-ssm-region.s3.amazonaws.com/latest/windows_amd64/  
AmazonSSMAgentSetup.exe
```

##代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 US East (Ohio) Region。有关受支持**##**值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#)的区域列。

3. 使用 Windows 服务控制面板或者通过在 PowerShell 中发送以下命令，来启动或重新启动 SSM 代理 (AmazonSSMAgent.exe)：

```
Restart-Service AmazonSSMAgent
```

Important

SSM 代理需要 Windows PowerShell 3.0 或更高版本才能在 Windows 实例上运行特定 SSM 文档 (例如，AWS-ApplyPatchBaseline 文档)。验证您的 Windows 实例是否在 Windows 管理框架 3.0 或更高版本上运行。此框架包括 Windows PowerShell。有关更多信息，请参阅 [Windows 管理框架 3.0](#)。

在 Windows 实例上查看 SSM 代理日志

SSM 代理将有关执行、计划操作、错误和运行状况的信息写入每个实例上的日志文件。您可以通过手动连接到实例，也可以将日志自动发送到 Amazon CloudWatch Logs，从而查看日志文件。有关将日志发送到 CloudWatch 的更多信息，请参阅[使用 AWS Systems Manager 监控实例 \(p. 389\)](#)。

您可以在以下位置中查看 Windows 实例的 SSM 代理日志文件。

- `%PROGRAMDATA%\Amazon\SSM\Logs\amazon-ssm-agent.log`
- `%PROGRAMDATA%\Amazon\SSM\Logs\errors.log`

配置 SSM 代理以使用适用于 Windows 实例的代理

本主题中的信息适用于在 2016 年 11 月或之后创建的不使用 Nano 安装选项的 Windows Server 实例。

如果实例是在 2016 年 11 月之前创建的 Windows Server 2003-2012 R2 实例，则 EC2Config 在实例上处理 Systems Manager 请求。有关配置 EC2Config 以使用代理的信息，请参阅[配置 EC2Config 服务的代理设置](#)。

对于使用 Nano 安装选项 (Nano Server) 的 Windows Server 2016 实例，您必须使用 PowerShell 连接。有关更多信息，请参阅[连接到 Windows Server 2016 Nano Server 实例](#)。

配置 SSM 代理以使用代理

1. 使用远程桌面或 Windows PowerShell，连接到您希望配置的实例，以使用代理。
2. 在 PowerShell 中运行以下命令块。将 `hostname` 和 `port` 替换为有关代理的信息：

```
$serviceKey = "HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent"  
$keyInfo = (Get-Item -Path $serviceKey).GetValue("Environment")  
$proxyVariables = @"http_proxy=hostname:port", "no_proxy=169.254.169.254"  
  
If($keyInfo -eq $null)  
{  
  New-ItemProperty -Path $serviceKey -Name Environment -Value $proxyVariables -  
  PropertyType MultiString -Force  
} else {
```

```
Set-ItemProperty -Path $serviceKey -Name Environment -Value $proxyVariables
}
Restart-Service AmazonSSMAgent
```

重置 SSM 代理的代理配置

1. 使用远程桌面或 Windows PowerShell，连接到要配置的实例。
2. 如果使用远程桌面连接，则以管理员的身份启动 PowerShell。
3. 在 PowerShell 中运行以下命令块：

```
Remove-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent -Name
Environment
Restart-Service AmazonSSMAgent
```

在 Linux 实例上安装和配置 SSM 代理

SSM 代理负责处理 Systems Manager 请求并按照请求中指定的方式配置计算机。使用以下过程可安装、配置或卸载 SSM 代理。

Important

- 在 2017.09 及以后日期版本的 Amazon Linux 基本 AMI 上，默认情况下会安装 SSM 代理。默认情况下，Amazon Linux 2、Ubuntu Server 16.04 和 Ubuntu Server 18.04 LTS AMI 上也会安装 SSM 代理。
- 对于其他版本的 Linux (包括经 Amazon ECS 优化的 AMI 等非基本映像)，您必须手动安装 SSM 代理。

在 [GitHub](#) 上可以找到 SSM 代理的源代码，您可以调整代理以满足您的需求。我们鼓励您针对要包含的更改提交 [提取请求](#)。但是，AWS 当前不支持运行此软件的修改后副本。

Note

要查看有关不同版本 SSM 代理的详细信息，请参阅[发行说明](#)。

主题

- [在 Amazon EC2 Linux 实例上手动安装 SSM 代理 \(p. 17\)](#)
- [配置 SSM 代理以使用代理 \(p. 26\)](#)
- [查看 SSM 代理日志 \(p. 28\)](#)
- [从 Linux 实例卸载 SSM 代理 \(p. 28\)](#)

在 Amazon EC2 Linux 实例上手动安装 SSM 代理

使用以下脚本之一在下列 Linux 实例上安装 SSM 代理。

- [Amazon Linux 和 Amazon Linux 2 \(p. 18\)](#)
- [Ubuntu Server \(p. 19\)](#)
- [Red Hat Enterprise Linux \(RHEL\) \(p. 22\)](#)
- [CentOS \(p. 23\)](#)
- [SUSE Linux Enterprise Server \(SLES\) 12 \(p. 24\)](#)
- [Raspbian \(p. 25\)](#)

使用以下脚本中的 URL 可从任意 AWS 区域下载 SSM 代理。如果您要从特定区域下载代理，请参阅[从特定区域下载 SSM 代理 \(p. 25\)](#)。

手动安装 SSM 代理后，在新版本可用时，您可以使用 Systems Manager 状态管理器 在您的实例上自动更新 SSM 代理。有关更多信息，请参阅 [演练：自动更新 SSM 代理 \(CLI\) \(p. 286\)](#)。

Amazon Linux 和 Amazon Linux 2

连接到您的 Amazon Linux 或 Amazon Linux 2 实例并执行下列步骤来安装 SSM 代理。使用 Systems Manager 对每个将运行命令的实例执行这些步骤。

Important

- 在 2017.09 及以后日期版本的 Amazon Linux 基本 AMI 上，默认情况下会安装 SSM 代理。默认情况下，Amazon Linux 2 AMI 上也会安装 SSM 代理。
- 对于其他版本的 Linux (包括经 Amazon ECS 优化的 AMI 等非基本映像)，您必须手动安装 SSM 代理。
- 从 Amazon Linux AMI 创建的使用代理的实例必须运行 Python `requests` 模块的当前版本，才能支持 Patch Manager 操作。有关更多信息，请参阅 [在使用代理服务器的 Amazon Linux 实例上升级 Python 请求模块 \(p. 28\)](#)。

在 Amazon Linux 或 Amazon Linux 2 上安装 SSM 代理

1. 在实例上创建临时目录。

```
mkdir /tmp/ssm
```

2. 更改为临时目录。

```
cd /tmp/ssm
```

3. 使用以下命令之一下载和运行 SSM 安装程序。

64 位实例：

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
```

32 位实例：

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_386/amazon-ssm-agent.rpm
```

4. 运行以下命令确定 SSM 代理是否在运行。该命令应返回“amazon-ssm-agent is running”消息。

Amazon Linux

```
sudo status amazon-ssm-agent
```

Amazon Linux 2

```
sudo systemctl status amazon-ssm-agent
```

5. 如果上一条命令返回“amazon-ssm-agent is stopped”消息，则运行以下命令。
 - a. 启动服务。

Amazon Linux

```
sudo start amazon-ssm-agent
```

Amazon Linux 2

```
sudo systemctl enable amazon-ssm-agent
```

```
sudo systemctl start amazon-ssm-agent
```

- b. 检查代理的状态。

Amazon Linux

```
sudo status amazon-ssm-agent
```

Amazon Linux 2

```
sudo systemctl status amazon-ssm-agent
```

Ubuntu Server

连接到您的 Ubuntu Server 实例，并执行下列过程之一中的步骤，在每个将使用 Systems Manager 运行命令的实例上安装 SSM 代理。

主题

- 关于 64 位 Ubuntu Server 16.04 实例上的 SSM 代理安装 (p. 19)
- 将 SSM 代理安装到 Ubuntu Server 18.04 和 16.04 LTS 64 位实例 (带 Snap 程序包) 上 (p. 20)
- 将 SSM 代理安装到 Ubuntu Server 16.04 和 14.04 64 位实例 (带 deb 安装程序包) 上 (p. 21)
- 将 SSM 代理安装到 Ubuntu Server 16.04 和 14.04 32 位实例上 (p. 21)

关于 64 位 Ubuntu Server 16.04 实例上的 SSM 代理安装

从通过使用 20180627 标识的 Ubuntu Server 16.04 AMI 创建的实例开始，已使用 Snap 程序包预安装 SSM 代理。例如：ubuntu/images/hvm-ssd/ubuntu-xenial-16.04-amd64-server-20180627。在通过以前的 AMI 创建的实例上，您应继续使用 deb 安装程序包。

Important

请注意，如果实例上已安装多个 SSM 代理 (例如，一个 SSM 代理是使用 Snap 安装的，一个 SSM 代理是使用 deb 安装程序安装的)，则代理操作将无法正常工作。

您可通过执行以下步骤来检查实例的源 AMI ID：

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. 在左侧导航窗格中，选择 Instances。
3. 选择一个实例。
4. 在 Description (描述) 选项卡上，在 AMI ID 字段中找到值。

对于从 64 位 Ubuntu Server 16.04 AMI 创建的实例，请确保遵循 SSM 代理安装类型的正确过程：

- 从带有标识符 **20180627** 的 AMI 或更高版本的 AMI 创建的实例：将 SSM 代理安装到 [Ubuntu Server 18.04 和 16.04 LTS 64 位实例 \(带 Snap 程序包\)](#) 上 (p. 20)
- 从 **20180627** 之前的 AMI 创建的实例：将 SSM 代理安装到 [Ubuntu Server 16.04 和 14.04 64 位实例 \(带 deb 安装程序包\)](#) 上 (p. 21)

将 SSM 代理安装到 Ubuntu Server 18.04 和 16.04 LTS 64 位实例 (带 Snap 程序包) 上

1. 默认情况下，SSM 代理安装到 Ubuntu Server 18.04 和 16.04 LTS 64 位 AMI (带标识符 20180627) 或更高版本上。有关版本 16.04 AMI 的更多信息，请参阅[关于 64 位 Ubuntu Server 16.04 实例上的 SSM 代理安装](#) (p. 19)。

如果您需要在本地服务器上安装 SSM 代理或者需要重新安装代理，您可以使用以下脚本。您无需为下载指定 URL，因为 snap 命令会自动从 [Snap 应用商店](https://snapcraft.io) (<https://snapcraft.io>) 下载代理。

```
sudo snap install amazon-ssm-agent --classic
```

Note

请注意 Ubuntu Server 18.04 和 16.04 上的 SSM 代理的以下详细信息：

- 由于 Snap 的已知问题，使用 snap 命令时您可能会看到 Maximum timeout exceeded 错误。如果您收到此错误，请运行以下命令（一次运行一条命令）来启动代理、停止它并检查其状态：

```
systemctl start snap.amazon-ssm-agent.amazon-ssm-agent.service
```

```
systemctl stop snap.amazon-ssm-agent.amazon-ssm-agent.service
```

```
systemctl status snap.amazon-ssm-agent.amazon-ssm-agent.service
```

- 在 Ubuntu Server 18.04 和 16.04 上，SSM 代理 安装程序文件（包括代理二进制文件和配置文件）存储在以下目录中：`/snap/amazon-ssm-agent/current/`。如果您更改配置文件 (`amazon-ssm-agent.json.template` 和 `seelog.xml.template`)，则必须将这些文件从 `/snap` 文件夹复制到 `/etc/amazon/ssm/` 文件夹。日志和库文件未更改 (`/var/lib/amazon/ssm`，`/var/log/amazon/ssm`)。
 - 在 Ubuntu Server 18.04 上，仅使用 Snaps。不要安装 deb 程序包。另外请确保实例上只安装并运行了代理的一个实例。
 - 在 Ubuntu Server 16.04 上，使用 Snap 或 deb 安装程序包安装 SSM 代理，具体取决于 16.04 AMI 的版本。有关更多信息，请参阅 [关于 64 位 Ubuntu Server 16.04 实例上的 SSM 代理安装](#) (p. 19)。
2. 运行以下命令确定 SSM 代理是否在运行。

```
sudo snap list amazon-ssm-agent
```

3. 如果上一条命令返回 `amazon-ssm-agent is stopped、inactive` 或 `disabled`，则运行以下命令将启动服务。

```
sudo snap start amazon-ssm-agent
```

4. 检查代理的状态。

```
sudo snap services amazon-ssm-agent
```

将 SSM 代理安装到 Ubuntu Server 16.04 和 14.04 64 位实例 (带 deb 安装程序包) 上

1. 如果您需要在本地服务器上安装 SSM 代理或者需要重新安装代理，您可以使用以下脚本。

Important

默认情况下，SSM 代理安装到从 Ubuntu Server 16.04 LTS 64 位 AMI (带标识符 20180627) 或更高版本上。从带更小标识符 (例如 20171121.1 和 20180522) 的 AMI 创建的实例将继续使用 deb 安装程序。

如果 SSM 代理与 Snap 一起安装到实例上，并且您使用 deb 安装程序包安装或更新 SSM 代理，则安装或 SSM 代理操作可能失败。有关更多信息，请参阅 [关于 64 位 Ubuntu Server 16.04 实例上的 SSM 代理安装 \(p. 19\)](#)。

在实例上创建临时目录。

```
mkdir /tmp/ssm
```

执行下面的命令。

```
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/  
amazon-ssm-agent.deb  
sudo dpkg -i amazon-ssm-agent.deb
```

2. 运行以下命令确定 SSM 代理是否在运行。

```
sudo systemctl status amazon-ssm-agent
```

3. 如果上一条命令返回 `amazon-ssm-agent is stopped`、`inactive` 或 `disabled`，则运行以下命令将启动服务。

```
sudo systemctl enable amazon-ssm-agent
```

4. 检查代理的状态。

```
sudo systemctl status amazon-ssm-agent
```

将 SSM 代理安装到 Ubuntu Server 16.04 和 14.04 32 位实例上

1. 在实例上创建临时目录。

```
mkdir /tmp/ssm
```

执行下面的命令。

```
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_386/amazon-  
ssm-agent.deb  
sudo dpkg -i amazon-ssm-agent.deb
```

2. 运行以下命令确定 SSM 代理是否在运行：

```
sudo status amazon-ssm-agent
```

3. 如果上一条命令返回 `amazon-ssm-agent is stopped`、`inactive` 或 `disabled`，则运行以下命令。

- a. 启动代理：

```
sudo start amazon-ssm-agent
```

- b. 检查代理的状态：

```
sudo status amazon-ssm-agent
```

Red Hat Enterprise Linux (RHEL)

连接到您的 RHEL 实例并执行下列步骤来安装 SSM 代理。使用 Systems Manager 对每个将运行命令的实例执行这些步骤。

在 Red Hat Enterprise Linux 上安装 SSM 代理

1. 在实例上创建临时目录。

```
mkdir /tmp/ssm
```

2. 使用以下命令之一下载和运行 SSM 安装程序。

64 位实例：

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
```

32 位实例：

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_386/amazon-ssm-agent.rpm
```

3. 运行以下命令之一以确定 SSM 代理是否在运行。该命令应返回“amazon-ssm-agent is running”消息。

RHEL 7.x:

```
sudo systemctl status amazon-ssm-agent
```

RHEL 6.x:

```
sudo status amazon-ssm-agent
```

4. 如果上一条命令返回“amazon-ssm-agent is stopped”，则运行以下命令。

- a. 启动服务。

RHEL 7.x:

```
sudo systemctl enable amazon-ssm-agent
```

```
sudo systemctl start amazon-ssm-agent
```

RHEL 6.x:

```
sudo start amazon-ssm-agent
```

- b. 检查代理的状态。

RHEL 7.x:

```
sudo systemctl status amazon-ssm-agent
```

RHEL 6.x:

```
sudo status amazon-ssm-agent
```

CentOS

连接到您的 CentOS 实例并执行下列步骤来安装 SSM 代理。使用 Systems Manager 对每个将运行命令的实例执行这些步骤。

在 CentOS 上安装 SSM 代理

1. 在实例上创建临时目录。

```
mkdir /tmp/ssm
```

2. 使用以下命令之一下载和运行 SSM 安装程序。

64 位实例：

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
```

32 位实例：

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_386/amazon-ssm-agent.rpm
```

3. 运行以下命令之一以确定 SSM 代理是否在运行。该命令应返回“amazon-ssm-agent is running”消息。

CentOS 7.x:

```
sudo systemctl status amazon-ssm-agent
```

CentOS 6.x:

```
sudo status amazon-ssm-agent
```

4. 如果上一条命令返回“amazon-ssm-agent is stopped”，则运行以下命令。

- a. 启动服务。

CentOS 7.x:

```
sudo systemctl enable amazon-ssm-agent
```

```
sudo systemctl start amazon-ssm-agent
```

CentOS 6.x:

```
sudo start amazon-ssm-agent
```

- b. 检查代理的状态。

CentOS 7.x:

```
sudo systemctl status amazon-ssm-agent
```

CentOS 6.x:

```
sudo status amazon-ssm-agent
```

SUSE Linux Enterprise Server (SLES) 12

连接到您的 SLES 实例并执行下列步骤来安装 SSM 代理。使用 Systems Manager 对每个将运行命令的实例执行这些步骤。

在 SUSE Linux Enterprise Server 上安装 SSM 代理

1. 在实例上创建临时目录。

```
mkdir /tmp/ssm
```

2. 更改为临时目录。

```
cd /tmp/ssm
```

3. 使用以下命令下载和运行 SSM 安装程序。

64 位实例：

```
wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm  
sudo rpm --install amazon-ssm-agent.rpm
```

4. 运行以下命令确定 SSM 代理是否在运行。该命令应返回“amazon-ssm-agent is running”消息。

```
sudo systemctl status amazon-ssm-agent
```

5. 如果上一条命令返回“amazon-ssm-agent is stopped”消息，则运行以下命令。

- a. 启动服务。

```
sudo systemctl enable amazon-ssm-agent  
sudo systemctl start amazon-ssm-agent
```

- b. 检查代理的状态。

```
sudo systemctl status amazon-ssm-agent
```

Raspbian

本部分包括有关如何在 Raspbian Jessie 和 Raspbian Stretch (包括 Raspberry Pi (32 位) 设备) 上安装 SSM 代理的信息。

开始前的准备工作

要将 Raspbian 设备设置为 Systems Manager 托管实例，您需要创建托管实例激活。完成激活后，您将收到一个激活代码和 ID。此代码/ID 组合具有 Amazon EC2 访问 ID 和私有密钥的功能，可提供从托管实例对 Systems Manager 服务的安全访问。在安全位置存储激活代码和 ID。有关激活流程的更多信息，请参阅[在混合环境中设置 AWS Systems Manager \(p. 31\)](#)。

连接到您的 Raspbian 设备并执行下列步骤来安装 SSM 代理。使用 Systems Manager 对每个将运行命令的实例执行这些步骤。

在 Raspbian 设备上安装 SSM 代理

1. 在实例上创建临时目录。

```
mkdir /tmp/ssm
```

2. 使用以下命令下载和运行 SSM 安装程序。

```
sudo curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_arm/  
amazon-ssm-agent.deb -o /tmp/ssm/amazon-ssm-agent.deb
```

3. 运行以下命令安装 SSM 代理：

```
sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
```

4. 运行以下命令停止 SSM 代理。

```
sudo service amazon-ssm-agent stop
```

5. 运行以下命令，以使用您完成托管实例激活过程时收到的托管实例激活代码和 ID 注册代理。

```
sudo amazon-ssm-agent -register -code "code" -id "ID" -region "region"
```

6. 运行以下命令启动 SSM 代理。

```
sudo service amazon-ssm-agent start
```

Note

如果在 SSM 代理错误日志中看到以下错误，说明计算机 ID 在重启后发生变更：

```
Unable to load instance associations, unable to retrieve  
associations unable to retrieve associations error occurred in  
RequestManagedInstanceRoleToken: MachineFingerprintDoesNotMatch:  
Fingerprint does not match
```

运行以下命令使计算机 ID 在重启后保持不变。

```
umount /etc/machine-id  
systemd-machine-id-setup
```

从特定区域下载 SSM 代理

如果要从某特定区域下载该代理，请复制操作系统的 URL，然后将 **region** 替换为适当的值。

##代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 US East (Ohio) Region。有关受支持**##**值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#)的区域列。

例如，要从美国西部 1 区域下载适用于 Amazon Linux、RHEL、CentOS 和 SLES 64 位的 SSM 代理，请使用以下 URL：

```
https://s3-us-west-1.amazonaws.com/amazon-ssm-us-west-1/latest/linux_amd64/amazon-ssm-agent.rpm
```

如果下载失败，请尝试用 `https://s3` 替换 `https://s3-##.##`。

- Amazon Linux、RHEL、CentOS 和 SLES 64 位：

```
https://s3-##.amazonaws.com/amazon-ssm-region/latest/linux_amd64/amazon-ssm-agent.rpm
```

- Amazon Linux、RHEL 和 CentOS 32 位：

```
https://s3-##.amazonaws.com/amazon-ssm-region/latest/linux_386/amazon-ssm-agent.rpm
```

- Ubuntu Server 64 位：

```
https://s3-##.amazonaws.com/amazon-ssm-region/latest/debian_amd64/amazon-ssm-agent.deb
```

- Ubuntu Server 32 位：

```
https://s3-##.amazonaws.com/amazon-ssm-region/latest/debian_386/amazon-ssm-agent.deb
```

- Raspbian：

```
https://s3-region.amazonaws.com/amazon-ssm-region/latest/debian_arm/amazon-ssm-agent.deb
```

配置 SSM 代理以使用代理

您可以将 `http_proxy`、`https_proxy` 和 `no_proxy` 设置添加到 `amazon-ssm-agent.override` 配置文件中，从而将 SSM 代理配置为通过 HTTP 通信。如果您安装 SSM 代理的较新或较早版本，覆盖文件还会保留代理设置。本部分包含适用于 upstart 和 systemd 环境的过程。

Note

从 Amazon Linux AMI 创建的使用代理的实例必须运行 Python `requests` 模块的当前版本，才能支持 Patch Manager 操作。有关更多信息，请参阅 [在使用代理服务器的 Amazon Linux 实例上升级 Python 请求模块](#) (p. 28)。

主题

- [配置 SSM 代理以使用代理 \(Upstart\)](#) (p. 26)
- [配置 SSM 代理以使用代理 \(systemd\)](#) (p. 27)
- [在使用代理服务器的 Amazon Linux 实例上升级 Python 请求模块](#) (p. 28)

配置 SSM 代理以使用代理 (Upstart)

1. 连接到已安装 SSM 代理的实例。
2. 打开一个简单编辑器 (例如 VIM) 并指定以下设置：

```
env http_proxy=http://hostname:port  
env https_proxy=https://hostname:port  
env no_proxy=169.254.169.254
```

Note

您必须添加 `no_proxy` 设置到文件并指定此处列出的 IP 地址。它是 Systems Manager 的实例元数据终端节点。没有此 IP 地址，对 Systems Manager 的调用将失败。

3. 将文件另存为以下位置中的 `amazon-ssm-agent.override` : `/etc/init/`
4. 使用以下命令停止和重新启动 SSM 代理：

```
sudo stop amazon-ssm-agent
sudo start amazon-ssm-agent
```

Note

有关在 Upstart 环境中使用 `.override` 文件的更多信息，请参阅 [init : Upstart init 守护程序任务配置](#)。

配置 SSM 代理以使用代理 (systemd)

以下过程中的步骤介绍了如何配置 SSM 代理 以在 systemd 环境中使用代理。此过程中的某些步骤包含适用于 Ubuntu Server 实例 (通过使用 Snap 安装) 的明确说明。

1. 连接到已安装 SSM 代理的实例。
2. 执行下面的 命令：

```
systemctl edit amazon-ssm-agent
```

对于通过使用 snap 安装的 Ubuntu Server 实例，请执行以下命令：

```
systemctl edit snap.amazon-ssm-agent.amazon-ssm-agent
```

3. 指定以下设置：

```
[Service]
Environment="http_proxy=http://hostname:port"
Environment="https_proxy=https://hostname:port"
Environment="no_proxy=169.254.169.254"
```

Note

您必须添加 `no_proxy` 设置到文件并指定此处列出的 IP 地址。它是 Systems Manager 的实例元数据终端节点。没有此 IP 地址，对 Systems Manager 的调用将失败。

4. 保存您的更改。系统在 `amazon-ssm-agent.service.d` 文件夹中创建 `amazon-ssm-agent.override` 文件。
5. 使用以下命令重新启动 SSM 代理：

```
sudo systemctl stop amazon-ssm-agent
sudo systemctl daemon-reload
```

对于通过使用 snap 安装的 Ubuntu Server 实例，使用以下命令重新启动 SSM 代理：

```
systemctl start snap.amazon-ssm-agent.amazon-ssm-agent
```


Note

有关在系统化环境中使用 `.override` 文件的更多信息，请参阅[修改现有 Unit 文件](#)。

在使用代理服务器的 Amazon Linux 实例上升级 Python 请求模块

要修补使用代理以及由 Amazon Linux AMI 创建的实例，Patch Manager 需要在该实例上安装 Python `requests` 模块的新版本。我们建议始终升级到最新的发布版本。

要确保已安装 Python `requests` 模块的最新版本，请遵循以下步骤：

1. 登录 Amazon Linux 实例，或在 Run Command 中使用 AWS-RunShellScript SSM 文档，在实例上运行以下命令：

```
pip list | grep requests
```

- 如果模块已安装，请求会在与以下内容类似的响应中返回版本号：

```
requests (1.2.3)
```

- 如果模块未安装，请运行以下命令进行安装：

```
pip install requests
```

- 如果未安装 pip 本身，请运行以下命令进行安装：

```
sudo yum install -y python-pip
```

2. 如果模块已安装，但列出的版本早于 2.18.4 (例如上一步中显示的 1.2.3)，请运行以下命令升级到 Python `requests` 模块的最新版本：

```
pip install requests --upgrade
```

查看 SSM 代理日志

SSM 代理将有关执行、计划操作、错误和运行状况的信息写入每个实例上的日志文件。您可以通过手动连接到实例，也可以将日志自动发送到 Amazon CloudWatch Logs，从而查看日志文件。有关将日志发送到 CloudWatch 的更多信息，请参阅[使用 AWS Systems Manager 监控实例 \(p. 389\)](#)。

您可以在以下位置中查看 Linux 实例的 SSM 代理日志。

- `/var/log/amazon/ssm/amazon-ssm-agent.log`
- `/var/log/amazon/ssm/errors.log`

SSM 代理 `stderr` 和 `stdout` 文件将写入到以下目录：`/var/lib/amazon/ssm`。

您可通过更新 `seelog.xml` 文件来启用延长日志记录。配置文件的默认位置为：`/etc/amazon/ssm/seelog.xml`

有关 `cihub/seelog` 配置的更多信息，请参阅 GitHub 上的 [Seelog Wiki](#)。有关 `cihub/seelog` 配置的示例，请参阅 GitHub 上的 [cihub/seelog examples](#) 存储库。

从 Linux 实例卸载 SSM 代理

使用以下命令卸载 SSM 代理。

Amazon Linux、Amazon Linux 2、RHEL 和 CentOS

```
sudo yum erase amazon-ssm-agent -y
```

Ubuntu Server

```
sudo dpkg -r amazon-ssm-agent
```

SLES

```
sudo rpm --erase amazon-ssm-agent
```

订阅 SSM 代理通知

Amazon SNS 可在 SSM 代理的新版本发布时向您发送通知。使用以下过程订阅这些通知。

订阅 SSM 代理通知

1. Open the Amazon SNS console at <https://console.aws.amazon.com/sns/v2/home>.
2. 在导航栏中，将区域更改为 US East (N. Virginia) (如果尚未选择)。您必须选择此区域，因为您订阅的 SNS 通知是在此区域中创建的。
3. 在导航窗格中，选择 Subscriptions。
4. 选择 Create subscription。
5. 在 Create subscription 对话框中，执行以下操作：
 - a. 对于 Topic ARN，请使用以下 Amazon 资源名称 (ARN)：

```
arn:aws:sns:us-east-1:720620558202:SSM-Agent-Update
```
 - b. 对于协议，选择 Email 或 SMS。
 - c. 对于 Endpoint，键入可用于接收通知的电子邮件地址。如果您选择 SMS，请键入区号和号码。
 - d. 选择 Create subscription。
6. 如果您选择了 Email，则会收到要求您确认订阅的电子邮件。打开电子邮件，然后按照说明操作以完成订阅。

当 SSM 代理的新版本发布时，我们会向订户发送通知。如果您不希望再收到这些通知，请通过以下步骤取消订阅。

取消订阅 SSM 代理通知

1. 打开 Amazon SNS 控制台。
2. 在导航窗格中，选择订阅。
3. 选择订阅，然后选择操作、删除订阅。在提示确认时，选择删除。

SSM 代理的最低 S3 存储桶权限

本主题提供有关资源为执行 Systems Manager 操作而可能需要访问的 Amazon Simple Storage Service (Amazon S3) 存储桶的信息。如果您想将实例配置文件或 VPC 终端节点的 Amazon S3 存储桶访问权限限制为使用 Systems Manager 所需的最小权限，则可在自定义策略中指定这些存储桶。

这些权限仅提供对 SSM 代理所需的资源的访问权限。它们不反映其他与 Amazon S3 相关的操作所需的权限。

内容

- [所需权限 \(p. 30\)](#)
- [示例 \(p. 30\)](#)

所需权限

下表描述了使用 Systems Manager 所需的每个 Amazon S3 存储桶策略权限。

SSM 代理所需的 Amazon S3 权限

许可	描述
<code>arn:aws:s3:::aws-ssm-<i>region</i>/*</code>	提供对 Amazon S3 存储桶的访问权限，该存储桶包含与 SSM 文档结合使用所需的模块。
<code>arn:aws:s3:::aws-windows-downloads-<i>region</i>/*</code>	一些支持 Windows 操作系统的 SSM 文档所必需的。
<code>arn:aws:s3:::amazon-ssm-packages-<i>region</i>/*</code>	使用 2.2.45.0 之前的版本的 SSM 代理运行文档 AWS-ConfigureAWSPackage 所必需的。
<code>arn:aws:s3:::<i>region</i>-birdwatcher-prod/*</code>	提供对由版本 2.2.45.0 或更高版本的 SSM 代理使用的分发服务的访问权限。此服务用于运行文档 AWS-ConfigureAWSPackage。

*##*代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 US East (Ohio) Region。有关受支持 *##* 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

示例

以下示例说明如何提供对 US East (Ohio) Region (`us-east-2`) 中的 Systems Manager 操作所需的 Amazon S3 存储桶的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": [
        "arn:aws:s3:::aws-ssm-us-east-2/*",
        "arn:aws:s3:::aws-windows-downloads-us-east-2/*",
        "arn:aws:s3:::amazon-ssm-packages-us-east-2/*",
        "arn:aws:s3:::us-east-2-birdwatcher-prod/*"
      ]
    }
  ]
}
```

Important

建议您避免使用通配符 (*) 来替代此策略中的特定区域，例如使用 `arn:aws:s3:::aws-ssm-*/` 替代 `arn:aws:s3:::aws-ssm-us-east-2/*`。这样做可以提供对您不打算授予权限的其他 Amazon S3 存储桶的访问权限。

在混合环境中设置 AWS Systems Manager

您可以借助 AWS Systems Manager 在混合环境中以远程方式安全地管理本地服务器和虚拟机 (VM)。为 Systems Manager 配置混合环境有以下优势。

- 创建一种一致且安全的方式，使用相同的工具或脚本从一个位置远程管理本地工作负载。
- 通过使用 AWS Identity and Access Management (IAM) 集中管理可在服务器和虚拟机上执行的操作的访问控制。
- 集中审计，您可以查看在您服务器和虚拟机上执行的操作，因为所有这些操作记录在 AWS CloudTrail 中。
- 集中监控，因为您可以配置 CloudWatch Events 和 Amazon SNS 以发送有关服务执行成功的通知。

完成本节中的过程来为 Systems Manager 配置混合计算机。

Important

完成后，您为 Systems Manager 配置的混合计算机会列在 Amazon EC2 控制台中，并被称为托管实例。为 Systems Manager 配置的 Amazon EC2 实例也是托管实例。但在 Amazon EC2 控制台中，您的本地实例与具有前缀“mi-”的 Amazon EC2 实例有别。

主题

- [为混合环境创建 IAM 服务角色 \(p. 31\)](#)
- [为混合环境创建托管实例激活 \(p. 32\)](#)
- [在 Windows 混合环境中的服务器和虚拟机上安装 SSM 代理 \(p. 34\)](#)
- [在 Linux 混合环境中的服务器和虚拟机上安装 SSM 代理 \(p. 35\)](#)

为混合环境创建 IAM 服务角色

混合环境中的服务器和虚拟机需要有 IAM 角色才能与 Systems Manager SSM 服务通信。该角色向 AssumeRole 授予对 SSM 服务的信任。

Note

您只需为每个 AWS 账户创建该服务角色一次。

使用 AWS Tools for Windows PowerShell 创建 IAM 服务角色

1. 使用以下信任策略创建文本文件 (在此示例中，它名为 `SSMService-Trust.json`)。使用 `.json` 文件扩展名保存该文件。

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

2. 使用 [New-IAMRole](#) (如下所示) 创建服务角色。此示例创建一个名为 `SSMServiceRole` 的角色。

```
New-IAMRole -RoleName SSMServiceRole -AssumeRolePolicyDocument (Get-Content -
raw SSMService-Trust.json)
```

3. 使用 [Register-IAMRolePolicy](#) (如下所示) 以允许 `SSMSERVICE_ROLE` 创建会话令牌。此会话令牌向托管实例授予使用 Systems Manager 运行命令的权限。

```
Register-IAMRolePolicy -RoleName SSMSERVICE_ROLE -PolicyArn arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM
```

使用 AWS CLI 创建 IAM 服务角色

1. 使用以下信任策略创建文本文件 (在此示例中, 它名为 `SSMSERVICE_ROLE-Trust.json`)。使用 `.json` 文件扩展名保存该文件。

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

2. 使用 [create-role](#) 命令创建服务角色。此示例创建一个名为 `SSMSERVICE_ROLE` 的角色。

```
aws iam create-role --role-name SSMSERVICE_ROLE --assume-role-policy-document file://SSMSERVICE_ROLE-Trust.json
```

3. 使用 [attach-role-policy](#) (如下所示) 以允许 `SSMSERVICE_ROLE` 创建会话令牌。此会话令牌向托管实例授予使用 Systems Manager 运行命令的权限。

```
aws iam attach-role-policy --role-name SSMSERVICE_ROLE --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM
```

Note

您的公司或组织中在混合计算机上使用 Systems Manager 的用户必须在 IAM 中获得授权, 才能调用 SSM API。有关更多信息, 请参阅 [配置对 Systems Manager 的访问权限](#) (p. 9)。

为混合环境创建托管实例激活

要在混合环境中将服务器和虚拟机设置为托管实例, 您需要创建托管实例激活。完成激活后, 您将收到一个激活代码和激活 ID。此代码/ID 组合具有 Amazon EC2 访问 ID 和私有密钥的功能, 可提供从托管实例对 Systems Manager 服务的安全访问。

创建托管实例激活 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中, 选择 Activations。

-或者-

如果 AWS Systems Manager 主页首先打开, 请选择菜单图标 (≡) 以打开导航窗格, 然后选择 Activations。

3. 选择创建激活。
4. (可选) 在激活说明字段中, 输入对此激活的描述。描述是可选的, 但如果要激活大量服务器和虚拟机, 建议输入描述。

- 在实例限制字段中，指定要向 AWS 注册的服务器或虚拟机的总数。
- 在 IAM 角色名称部分中，选择一个服务角色选项，以便服务器和虚拟机能够与云中的 AWS Systems Manager 进行通信：
 - 选择使用系统创建的默认命令执行角色使用 AWS 创建的角色和托管策略。
 - 选择一个具有所需权限的现有自定义 IAM 角色使用先前创建的可选自定义角色。
- 在激活过期日期字段中，为激活指定到期日期。

Note

如果需要在过期日期后注册更多的托管实例，您必须创建新的激活。过期日期对已注册且正在运行的实例没有任何影响。

- (可选) 在默认实例名称字段中指定名称。
- 选择创建激活。

Important

在安全位置存储托管实例激活代码和激活 ID。在混合环境中的服务器和虚拟机上安装 SSM 代理时指定此代码和 ID。如果丢失了此代码和 ID，则必须创建一个新激活。

使用 AWS Tools for Windows PowerShell 创建托管实例激活

- 在安装了 AWS Tools for Windows PowerShell 的计算机上，在 AWS Tools for Windows PowerShell 中运行以下命令。

```
New-SSMActivation -DefaultInstanceName name -IamRole iam-service-role-name -  
RegistrationLimit number-of-managed-instances -Region region
```

*##*代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 us-east-2 表示 US East (Ohio) Region。有关受支持 *##* 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

例如：

```
New-SSMActivation -DefaultInstanceName MyWebServers -IamRole RunCommandServiceRole -  
RegistrationLimit 10 -Region us-east-2
```

- 按 Enter。如果激活成功，系统将返回一个激活代码和一个激活 ID。在安全位置存储激活代码和激活 ID。

使用 AWS CLI 创建托管实例激活

- 在安装了 AWS Command Line Interface (AWS CLI) 的计算机上，在 CLI 中运行以下命令。

```
aws ssm create-activation --default-instance-name name --iam-role IAM service role --  
registration-limit number of managed instances --region region
```

*##*代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 us-east-2 表示 US East (Ohio) Region。有关受支持 *##* 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

例如：

```
aws ssm create-activation --default-instance-name MyWebServers --iam-role  
RunCommandServiceRole --registration-limit 10 --region us-east-2
```

- 按 Enter。如果激活成功，系统将返回一个激活代码和一个激活 ID。在安全位置存储激活代码和激活 ID。

在 Windows 混合环境中的服务器和虚拟机上安装 SSM 代理

在开始前，找到在上一部分中完成托管实例激活后收到的激活码和激活 ID。按照下面的流程指定激活码和 ID。

Important

此过程适用于本地或混合环境中的服务器和 VM。要在 Amazon EC2 Windows 实例上下载并安装 SSM 代理，请参阅[在 Windows 实例上安装和配置 SSM 代理 \(p. 15\)](#)。

在混合环境中的服务器和虚拟机上安装 SSM 代理

- 登录混合环境中的服务器或虚拟机。
- 打开 Windows PowerShell。
- 将以下命令块复制并粘贴到 AWS Tools for Windows PowerShell 中。将占位符值替换为在创建托管实例激活时生成的激活代码和激活 ID，以及要从其下载 SSM 代理的 AWS 区域的标识符。

##代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 us-east-2 表示 US East (Ohio) Region。有关受支持**##**值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#)的区域列。

```
$code = "activation-code"
$id = "activation-id"
$region = "region"
$dir = $env:TEMP + "\ssm"
New-Item -ItemType directory -Path $dir -Force
cd $dir
(New-Object System.Net.WebClient).DownloadFile("https://amazon-ssm-
$region.s3.amazonaws.com/latest/windows_amd64/AmazonSSMAgentSetup.exe", $dir +
"\AmazonSSMAgentSetup.exe")
Start-Process .\AmazonSSMAgentSetup.exe -ArgumentList @("/q", "/log", "install.log",
"CODE=$code", "ID=$id", "REGION=$region") -Wait
Get-Content ($env:ProgramData + "\Amazon\SSM\InstanceData\registration")
Get-Service -Name "AmazonSSMAgent"
```

- 按 Enter。

此命令执行以下操作：

- 下载 SSM 代理并将它安装到服务器或虚拟机上。
- 将服务器或虚拟机注册到 SSM 服务。
- 返回类似以下内容的请求响应：

```
Directory: C:\Users\ADMINI-1\AppData\Local\Temp\2

Mode                LastWriteTime         Length Name
----                -
d-----          07/07/2018    8:07 PM             ssm
{"ManagedInstanceID":"mi-008d36be46EXAMPLE","Region":"us-east-2"}

Status              : Running
```



```
Name      : AmazonSSMAgent
DisplayName : Amazon SSM Agent
```

服务器或虚拟机现在为托管实例。在控制台中，这些实例在列出时带有前缀“mi-”。您可以使用 `List` 命令查看所有实例。有关更多信息，请参阅 [Amazon EC2 Systems Manager API Reference](#)。

在 Linux 混合环境中的服务器和虚拟机上安装 SSM 代理

在开始前，找到在完成托管实例激活后收到的激活码和激活 ID。按照下面的流程指定激活码和 ID。

Important

此过程适用于本地或混合环境中的服务器和 VM。要在 Amazon EC2 Linux 实例上下载并安装 SSM 代理，请参阅[在 Linux 实例上安装和配置 SSM 代理 \(p. 17\)](#)。

使用以下脚本中的 URL 可从任意 AWS 区域下载 SSM 代理。如果需要从某特定区域下载该代理，请复制操作系统的 URL，然后将 `region` 替换为适当的值。

`##`代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 US East (Ohio) Region。有关受支持 `##` 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

例如，要从 US West (N. California) Region (`us-west-1`) 下载适用于 Amazon Linux、RHEL、CentOS 和 SLES 64 位的 SSM 代理，请使用以下 URL：

```
https://s3-us-west-1.amazonaws.com/amazon-ssm-us-west-1/latest/linux_amd64/amazon-ssm-agent.rpm
```

如果下载失败，请尝试用 `https://s3` 替换 `https://s3-##`。 `##`。

- Amazon Linux、RHEL、CentOS 和 SLES 64 位

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_amd64/amazon-ssm-agent.rpm
```

- Amazon Linux、RHEL 和 CentOS 32 位

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/linux_386/amazon-ssm-agent.rpm
```

- Ubuntu Server 64 位

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_amd64/amazon-ssm-agent.deb
```

- Ubuntu Server 32 位

```
https://s3.region.amazonaws.com/amazon-ssm-region/latest/debian_386/amazon-ssm-agent.deb
```

- Raspbian

```
https://s3-region.amazonaws.com/amazon-ssm-region/latest/debian_arm/amazon-ssm-agent.deb
```

在混合环境中的服务器和虚拟机上安装 SSM 代理

1. 登录混合环境中的服务器或虚拟机。
2. 将以下命令块之一复制并粘贴到 SSH 中。将占位符值替换为在创建托管实例激活时生成的激活代码和激活 ID，以及要从其下载 SSM 代理的 AWS 区域的标识符。

请注意，如果您不是根用户，则 `sudo` 不是必需的。

##代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 `us-east-2` 表示 US East (Ohio) Region。有关受支持**##**值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#)的区域列。

在 Amazon Linux、RHEL 6.x 和 CentOS 6.x 上

```
mkdir /tmp/ssm
curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm -o /tmp/ssm/amazon-ssm-agent.rpm
sudo yum install -y /tmp/ssm/amazon-ssm-agent.rpm
sudo stop amazon-ssm-agent
sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
sudo start amazon-ssm-agent
```

在 RHEL 7.x 和 CentOS 7.x 上

```
mkdir /tmp/ssm
curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm -o /tmp/ssm/amazon-ssm-agent.rpm
sudo yum install -y /tmp/ssm/amazon-ssm-agent.rpm
sudo systemctl stop amazon-ssm-agent
sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
sudo systemctl start amazon-ssm-agent
```

在 SLES 上

```
mkdir /tmp/ssm
sudo wget https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
sudo rpm --install amazon-ssm-agent.rpm
sudo systemctl stop amazon-ssm-agent
sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
sudo systemctl enable amazon-ssm-agent
sudo systemctl start amazon-ssm-agent
```

在 Ubuntu 上

```
mkdir /tmp/ssm
curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_amd64/amazon-ssm-agent.deb -o /tmp/ssm/amazon-ssm-agent.deb
sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
sudo service amazon-ssm-agent stop
sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
sudo service amazon-ssm-agent start
```

在 Raspbian 上

```
mkdir /tmp/ssm
sudo curl https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/debian_arm/amazon-ssm-agent.deb -o /tmp/ssm/amazon-ssm-agent.deb
sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
sudo service amazon-ssm-agent stop
sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
```

```
sudo service amazon-ssm-agent start
```

Note

如果在 SSM 代理错误日志中看到以下错误，说明计算机 ID 在重启后发生变更：
Unable to load instance associations, unable to retrieve
associations unable to retrieve associations error occurred in
RequestManagedInstanceRoleToken: MachineFingerprintDoesNotMatch:
Fingerprint does not match
执行以下命令使计算机 ID 在重启后保持不变。

```
umount /etc/machine-id  
systemd-machine-id-setup
```

3. 按 Enter。

此命令将下载 SSM 代理并将它安装到混合环境中的服务器或虚拟机上。此命令停止 SSM 代理，然后向 SSM 服务注册服务器或虚拟机。服务器或虚拟机现在为托管实例。为 Systems Manager 配置的 Amazon EC2 实例也是托管实例。但在 Amazon EC2 控制台中，本地实例与具有前缀“mi-”的 Amazon EC2 实例有区别。

合作伙伴和产品集成

您可以将 AWS Systems Manager 与 GitHub、Amazon S3 和卷影复制服务 (VSS) 等合作伙伴技术和产品技术结合使用，从而自动完成托管实例的部署、配置和维护。

内容

- [从 GitHub 和 Amazon S3 运行脚本 \(p. 38\)](#)
- [使用 Run Command 为 EBS 卷拍摄启用 VSS 的快照 \(p. 47\)](#)
- [将 Chef InSpec 配置文件与 Systems Manager Compliance 结合使用 \(p. 55\)](#)

从 GitHub 和 Amazon S3 运行脚本

本部分介绍如何使用 `AWS-RunRemoteScript` 预定义的 SSM 文档下载来自 GitHub 和 Amazon S3 的脚本，包括 Ansible、Python、Ruby 和 PowerShell 脚本。通过使用该文档，不再需要将脚本手动导入 Amazon EC2，也不需要将其封装在 SSM 文档中。通过将 Systems Manager 与 GitHub 和 Amazon S3 集成，可以促进实现基础设施即代码，由此可缩短管理实例需要的时间，同时在队列中实现标准化配置。

还可以创建自定义 SSM 文档，实现从远程位置即可下载并运行脚本或其他 SSM 文档。有关更多信息，请参阅 [创建复合文档 \(p. 308\)](#)。

主题

- [从 GitHub 运行脚本 \(p. 38\)](#)
- [从 Amazon S3 运行脚本 \(p. 43\)](#)

从 GitHub 运行脚本

本部分介绍如何从私有或公有 GitHub 存储库下载并运行脚本。您可以运行不同类型的脚本，包括 Ansible Playbooks、Python、Ruby 和 PowerShell 脚本。

还可以下载包括多个脚本的目录。在运行目录中的主脚本时，Systems Manager 也会同时运行所有引用的脚本 (只要引用的脚本包含在该目录中)。

请注意关于从 GitHub 运行脚本的下列重要详细信息。

- Systems Manager 不检查您的脚本是否能够在实例上运行。在下载并运行脚本之前，必须验证对应实例上是否已经安装所需软件。您也可以创建一个复合文档，以使用 Run Command 或 State Manager 安装软件，然后再下载并运行脚本。
- 您负责确保满足所有 GitHub 要求，包括根据需要刷新访问令牌。另外，必须确保不超过已验证或未验证请求的数量。有关更多信息，请参阅 GitHub 文档。

主题

- [从 GitHub 运行 Ansible Playbooks \(p. 38\)](#)
- [从 GitHub 运行 Python 脚本 \(p. 41\)](#)

从 GitHub 运行 Ansible Playbooks

本部分包含帮助您使用 控制台或 AWS CLI 从 GitHub 运行 Ansible Playbooks 的过程。

开始前的准备工作

如果您计划运行私有 GitHub 存储库中存储的脚本，必须为 GitHub 安全访问令牌创建 Systems Manager SecureString 参数。通过 SSH 手动传递令牌无法访问私有 GitHub 存储库中的脚本。访问令牌必须作为 Systems Manager SecureString 参数传递。有关创建 SecureString 参数的更多信息，请参阅[创建 Systems Manager 参数 \(p. 369\)](#)。

从 GitHub (控制台) 运行 Ansible Playbook

从 GitHub 运行 Ansible Playbook

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择运行命令。
4. 在命令文档列表中，选择 AWS-RunRemoteScript。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. 在命令参数中，执行以下操作：
 - 在源类型中，选择 GitHub。
 - 在源信息框中，按以下格式键入访问源所需的信息：

```
{ "owner": "owner_name", "repository": "repository_name",  
  "path": "path_to_scripts_or_directory", "tokenInfo": "{ \"ssm-secure\":  
    SecureString_parameter_name }" }
```

例如：

```
{ "owner": "TestUser1", "repository": "GitHubPrivateTest", "path": "scripts/  
webserver.yml", "tokenInfo": "{ \"ssm-secure\": mySecureStringParameter }" }
```

此示例下载名为 complex-script 的脚本目录。

- 在 Command Line 字段中，键入脚本执行的参数。以下是示例。

```
ansible-playbook -i "localhost," --check -c local webserver.yml
```

- (可选) 在工作目录字段中，键入要在其中下载和运行脚本的实例上的目录名称。
 - (可选) 在执行超时中，指定脚本命令执行失败之前系统要等待的秒数。
7. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
 8. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
9. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

10. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

11. 选择 Run。

使用 AWS CLI 从 GitHub 运行 Ansible Playbook

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令可从 GitHub 下载并运行脚本。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "instance-IDs" --parameters '{"sourceType":["GitHub"],"sourceInfo":[{"owner\":"owner_name", "repository\":"repository_name", "path\":"path_to_file_or_directory", "tokenInfo\":"{{ssm-secure:name_of_your_SecureString_parameter}}"}],"commandLine":["commands_to_run"]}'
```

以下是示例。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "i-1234abcd" --parameters '{"sourceType":["GitHub"],"sourceInfo":[{"owner\":"TestUser1\", "repository\":"GitHubPrivateTest\", "path\":"scripts/webserver.yml\", "tokenInfo\":"{{ssm-secure:mySecureStringParameter}}"}],"commandLine":["ansible-playbook -i "localhost," --check -c local webserver.yml"]}'
```

从 GitHub 运行 Python 脚本

本部分包含帮助您使用控制台或 AWS CLI 从 GitHub 运行 Python 脚本的过程。

从 GitHub (控制台) 运行 Python 脚本

从 GitHub 运行 Python 脚本

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择运行命令。
4. 在命令文档列表中，选择 AWS-RunRemoteScript。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. 在命令参数中，执行以下操作：

- 在源类型中，选择 GitHub。
- 在源信息文本框中，按以下格式键入访问源所需的信息：

```
{ "owner": "owner_name", "repository": "repository_name",  
  "path": "path_to_scripts_or_directory", "tokenInfo": "{ {ssm-  
secure:SecureString_parameter_name } }" }
```

例如：

```
{ "owner": "TestUser1", "repository": "GitHubPrivateTest", "path": "scripts/python/  
complex-script", "tokenInfo": "{ {ssm-secure:mySecureStringParameter } }" }
```

此示例下载名为 complex-script 的脚本目录。

- 在 Command Line 字段中，键入脚本执行的参数。以下是示例。

```
mainFile.py argument-1 argument-2
```

该示例运行 mainFile.py，后者之后会运行 complex-script 目录中的其他脚本。

- (可选) 在工作目录字段中，键入要在其中下载和运行脚本的实例上的目录名称。
 - (可选) 在执行超时中，指定脚本命令执行失败之前系统要等待的秒数。
7. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
 8. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
9. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

10. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

11. 选择 Run。

使用 AWS CLI 从 GitHub 运行 Python 脚本

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令可从 GitHub 下载并运行脚本。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "instance-IDs" --parameters '{"sourceType":["GitHub"],"sourceInfo":[{"owner\":"owner_name", "repository\":"repository_name", "path\":"path_to_script_or_directory"}],"commandLine":["commands_to_run"]}'
```

以下是示例。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "i-abcd1234" --parameters '{"sourceType":["GitHub"],"sourceInfo":[{"owner\":"TestUser1", "repository\":"GitHubTestPublic", "path\":"scripts/python/complex-script"}],"commandLine":["mainFile.py argument-1 argument-2 "]}'
```

此示例下载名为 complex-script 的脚本目录。commandLine 条目运行 mainFile.py，它之后会运行 complex-script 目录中的其他脚本。

从 Amazon S3 运行脚本

本部分介绍如何从 Amazon S3 下载并运行脚本。您可以运行不同类型的脚本，包括 Ansible Playbooks、Python、Ruby 和 PowerShell。

还可以下载包括多个脚本的目录。在运行目录中的主脚本时，Systems Manager 也会同时运行所有引用的脚本（只要引用的脚本包含在该目录中）。

请注意关于从 Amazon S3 运行脚本的下列重要详细信息。

- Systems Manager 不检查您的脚本是否能够在实例上运行。在下载并运行脚本之前，必须验证对应实例上是否已经安装所需软件。您也可以创建一个复合文档，以使用 Run Command 或 State Manager 安装软件，然后再下载并运行脚本。
- 验证您的 AWS Identity and Access Management (IAM) 用户账户、角色或组是否有权读取 S3 存储桶。

主题

- [从 Amazon S3 运行 Ruby 脚本 \(p. 43\)](#)
- [从 Amazon S3 运行 PowerShell 脚本 \(p. 45\)](#)

从 Amazon S3 运行 Ruby 脚本

本部分包含帮助您使用 EC2 控制台或 AWS CLI 从 Amazon S3 运行 Ruby 脚本的过程。

从 Amazon S3 (控制台) 运行 Ruby 脚本

从 Amazon S3 运行 Ruby 脚本

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择运行命令。
4. 在命令文档列表中，选择 AWS-RunRemoteScript。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. 在命令参数中，执行以下操作：
 - 在源类型中，选择 S3。
 - 在源信息文本框中，按以下格式键入访问源所需的信息：

```
{"path":"https://s3.amazonaws.com/path_to_script"}
```

例如：

```
{"path":"https://s3.amazonaws.com/rubyttest/scripts/ruby/helloWorld.rb"}
```


- 在 Command Line 字段中，键入脚本执行的参数。以下是示例。

```
helloWorld.rb argument-1 argument-2
```

- (可选) 在工作目录字段中，键入要在其中下载和运行脚本的实例上的目录名称。
- (可选) 在执行超时中，指定脚本命令执行失败之前系统要等待的秒数。

7. 在 Other parameters 中：

- 在 Comment 框中，键入有关此命令的信息。
- 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。

8. (可选) 在 Rate control 中：

- 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。

9. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

10. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

11. 选择 Run。

使用 AWS CLI 从 S3 运行 Ruby 脚本

- 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

- 执行以下命令可从 Amazon S3 下载并运行脚本。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "instance-IDs" --parameters '{"sourceType":["S3"],"sourceInfo":[{"path":"https://s3.amazonaws.com/path_to_script"}],"commandLine":["script_name_and_arguments"]}'
```

以下是示例。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "i-abcd1234"
--parameters '{"sourceType":["S3"],"sourceInfo":[{"path\":"https://s3.amazonaws.com/
RubyTest/scripts/ruby/helloWorld.rb\"}]","commandLine":["helloWorld.rb argument-1
argument-2"]}'
```

从 Amazon S3 运行 PowerShell 脚本

本部分包含帮助您使用 EC2 控制台或 AWS CLI 从 Amazon S3 运行 PowerShell 脚本的过程。

从 Amazon S3 (控制台) 运行 PowerShell 脚本

从 Amazon S3 运行 PowerShell 脚本

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择运行命令。
4. 在命令文档列表中，选择 AWS-RunRemoteScript。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. 在命令参数中，执行以下操作：
 - 在源类型中，选择 S3。
 - 在源信息文本框中，按以下格式键入访问源所需的信息：

```
{"path": "https://s3.amazonaws.com/path_to_script"}
```

例如：

```
{"path": "https://s3.amazonaws.com/PowerShellTest/powershell/helloPowershell.ps1"}
```

- 在 Command Line 字段中，键入脚本执行的参数。以下是示例。

```
helloPowershell.ps1 argument-1 argument-2
```

- (可选) 在工作目录字段中，键入要在其中下载和运行脚本的实例上的目录名称。
 - (可选) 在执行超时中，指定脚本命令执行失败之前系统要等待的秒数。
7. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
 8. (可选) 在 Rate control 中：

- 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
9. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

10. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [为 Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

11. 选择 Run。

使用 AWS CLI 从 S3 运行 PowerShell 脚本

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令可从 Amazon S3 下载并运行脚本。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids "instance-IDs" --parameters '{"sourceType":["S3"],"sourceInfo":[{"path": "\https://s3.amazonaws.com/path_to_script\"}"],"commandLine":["script_name_and_arguments"]}'
```

以下是示例。

```
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-ids
  "i-1234abcd" --parameters '{"sourceType":["S3"],"sourceInfo":[{"path": "\https://
s3.amazonaws.com/TestPowershell/powershell/helloPowershell.ps1\"}"],"commandLine":
["helloPowershell.ps1 argument-1 argument-2"]}'
```

使用 Run Command 为 EBS 卷拍摄启用 VSS 的快照

使用 Run Command，您可以为附加到 Amazon EC2 Windows 实例的所有 [Amazon Elastic Block Store \(Amazon EBS\)](#) 卷拍摄应用程序一致性快照。快照过程使用 Windows [卷影复制服务 \(VSS\)](#) 对 VSS 感知应用程序进行映像级备份，包括这些应用程序和磁盘之间的待处理事务中的数据。此外，当需要备份所有已连接的卷时，无需关闭实例或断开与它们的连接。

使用启用 VSS 的 EBS 快照不会产生额外费用。您只需为备份过程创建的 EBS 快照付费。有关更多信息，请参阅[如何计算我的 EBS 快照账单？](#)

如何使用

下面介绍了拍摄启用 VSS 的应用程序一致性 EBS 快照的过程。

1. 您可以验证和配置 Systems Manager 先决条件。
2. 可以输入 AWSEC2-CreateVssSnapshot SSM 文档的参数并使用 Run Command 执行本文档。您无法为特定卷创建启用 VSS 的 EBS 快照。但是，您可以指定参数以便从备份过程中排除启动卷。
3. 您的实例上的 VSS 代理会协调正在运行的应用程序的所有正在进行的 I/O 操作。
4. 系统会刷新所有 I/O 缓冲区并临时暂停所有 I/O 操作。暂停最多持续 10 秒钟。
5. 在暂停期间，系统会为附加到实例的所有卷创建快照。
6. 暂停解除，I/O 恢复操作。
7. 系统将所有新建快照添加到 EBS 快照列表。系统会为此过程成功创建的所有启用 VSS 的 EBS 快照添加 AppConsistent:true 标签。此标签有助于将此过程创建的快照与其他过程创建的快照区分开来。如果系统遇到错误，此过程创建的快照则不会包含 AppConsistent:true 标签。
8. 在需要从快照还原的情况下，您可以使用从快照创建卷的标准 EBS 过程进行还原，也可以使用示例脚本将所有卷还原到实例 (本部分稍后将进行介绍)。

开始前的准备工作

在使用 Run Command 创建启用 VSS 的 EBS 快照之前，请检查以下要求和限制，并完成要求的任务。

- 运行 Windows Server 2008 R2 或更高版本的实例可支持启用 VSS 的 EBS 快照。(当前不支持 Windows Server 2008 R2 Core。)验证您的实例是否满足 Amazon EC2 Windows 的所有要求。有关更多信息，请参阅[设置 AWS Systems Manager \(p. 6\)](#)。
- 更新您的实例以使用 SSM 代理 版本 2.2.58.0 或更高版本。如果您使用的是 SSM 代理的较旧版本，可以使用 Run Command 来进行更新。有关更多信息，请参阅[示例：更新 SSM 代理 \(p. 183\)](#)。
- Systems Manager 需要在您的实例上执行操作的权限。您必须为每个实例配置适用于 Systems Manager 的 AWS Identity and Access Management (IAM) 实例配置文件角色。有关更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。
- Systems Manager 需要创建和标记启用 VSS 的 EBS 快照的权限。您可以配置启用这些权限的 IAM 角色。您必须为每个实例配置用于创建和标记快照的角色。有关更多信息，请参阅[设置任务中的为启用 VSS 的 EBS 快照创建 IAM 角色 \(p. 48\)](#)。

Note

如果您不希望向您的实例分配快照角色，可以使用 Run Command 和预定义的 AWSEC2-ManagedVssIO SSM 文档来临时暂停 I/O，创建启用 VSS 的 EBS 快照，并重新启动 I/O。此过程在执行命令的用户上下文中运行。如果用户具有足够的权限来创建和标记快照，那么 Systems Manager 可以创建和标记启用 VSS 的 EBS 快照，而实例上无需更多 IAM 快照角色。仍然必须为实例配置实例配置文件角色。有关更多信息，请参阅[通过使用 AWSEC2-ManagedVssIO SSM 文档 \(高级\) 创建启用 VSS 的 EBS 快照 \(p. 54\)](#)。

- Systems Manager 需要在您的实例上安装 VSS 组件。如果您需要安装必需的 VSS 组件，则可以下载适用于 Systems Manager 的 VSS 程序包并在您的实例上运行。如果您计划使用自己的 VSS (BYOL) 的 Microsoft VSS (BYOL) 许可，仍然需要安装适用于 Systems Manager 的 VSS 组件。有关更多信息，请参阅设置任务中的[下载并安装适用于 Systems Manager 的 VSS 组件](#) (p. 49)。

设置任务

- [为启用 VSS 的 EBS 快照创建 IAM 角色](#) (p. 48)
- [下载并安装适用于 Systems Manager 的 VSS 组件](#) (p. 49)

为启用 VSS 的 EBS 快照创建 IAM 角色

此部分包括创建 IAM 策略的步骤以及一个用于创建 IAM 角色的独立步骤，该 IAM 角色使用您在第一步中创建的策略。该策略使 Systems Manager 能够创建快照、标记快照，并将设备 ID 等元数据附加到系统创建的默认快照标签。

为启用 VSS 的快照创建 IAM 策略

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中选择 Policies，然后选择 Create policy。
3. 在创建策略页面上，选择 JSON 选项卡，然后复制以下 JSON 策略并粘贴到文本框中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:*::snapshot/*"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:CreateSnapshot"
      ],
      "Resource": "*"
    }
  ]
}
```

4. 在查看策略页面上，在策略名称字段中键入名称，然后选择创建策略。系统将返回 IAM 控制台。

使用以下过程为启用了 VSS 的快照创建 IAM 角色。此角色包括适用于 Amazon EC2 和 Systems Manager 的策略。

为启用 VSS 的 EBS 快照创建 IAM 角色

1. 在导航窗格中，选择 Roles，然后选择 Create Role。
2. 在 Select type of trusted entity 页面上的 AWS Service 下，选择 EC2。
3. 在选择您的使用案例部分，选择 EC2，然后选择下一步：权限。
4. 在 Attach permissions policy 页面中，选择您刚刚创建的策略，然后选择 Next: Review。
5. 在 Review 页面上，在 Role name 框中键入名称，然后键入描述。
6. 选择 Create role。系统将让您返回到 Roles 页。

7. 选择您刚刚创建的角色。角色的摘要页面打开。
8. 选择 Attach policy。
9. 搜索并选择 AmazonEC2RoleforSSM。
10. 选择 Attach policy。
11. 将此角色附加到要为其创建启用 VSS 的 EBS 快照的实例。有关更多信息，请参阅 Amazon EC2 用户指南 中的 [将 IAM 角色附加到实例](#)。

下载并安装适用于 Systems Manager 的 VSS 组件

Systems Manager 需要在您的实例上安装 VSS 组件。使用以下程序配置 状态管理器 关联，该关联可通过使用 AwsVssComponents 程序包自动下载并安装组件。当 AWS 发布新版本的程序包时，状态管理器 关联将自动下载并安装该程序包。该程序包由两个组件构成：VSS 请求程序和 VSS 提供程序。系统将这些组件复制到实例上的某个目录，然后将提供程序 DLL 注册为 VSS 提供程序。

如果您不希望 Systems Manager 在此程序包的新版本可用时自动下载并安装该程序包，可以使用 Run Command 和 AWS-ConfigureAWSPackage SSM 文档在实例上安装该程序包，如本部分中稍后所述。

创建可自动下载并安装 AwsVssComponents 程序包 () 的 状态管理器 关联

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 状态管理器。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 状态管理器。

3. 选择创建关联。
4. (可选) 在名称字段中，键入一个描述性名称。
5. 在命令文档列表中，选择 AWS-ConfigureAWSPackage。
6. 在参数部分的操作菜单中选择安装。
7. 在名称字段中，键入 AwsVssComponents。
8. 在版本字段中，验证是否自动填充了最新。
9. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅 [我的实例在哪里？ \(p. 202\)](#) 中的故障排除提示。

10. 在指定计划部分，选择一个选项。
11. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

12. 选择 Create Association，然后选择 Close。系统将尝试在实例上创建关联并立即应用状态。关联状态显示 Pending。
13. 在 Association 页面的右角，选择刷新按钮。如果您在一个或多个 EC2 Windows 实例上创建关联，状态将更改为 Success。如果没有为 Systems Manager 正确配置实例，状态将显示为失败。
14. 如果状态为失败，请验证 SSM 代理 是否在实例上运行，并验证实例是配置了适用于 Systems Manager 的 IAM 角色。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

使用 AWS CLI 安装 VSS 程序包

使用以下过程用 Run Command 从 AWS CLI 下载 AwsVssComponents 程序包并在您的实例上进行安装。该程序包由两个组件构成：VSS 请求程序和 VSS 提供程序。系统将这些组件复制到实例上的某个目录，然后将提供程序 DLL 注册为 VSS 提供程序。

使用 AWS CLI 安装 VSS 程序包

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令下载并安装适用于 Systems Manager 的 VSS 必需组件。

```
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids
  "i-12345678" --parameters '{"action":["Install"],"name":["AwsVssComponents"]}'
```

使用 Tools for Windows PowerShell 安装 VSS 程序包

使用以下过程用 Run Command 从 Tools for Windows PowerShell 下载 AwsVssComponents 程序包并在您的实例上进行安装。该程序包由两个组件构成：VSS 请求程序和 VSS 提供程序。系统将这些组件复制到实例上的某个目录，然后将提供程序 DLL 注册为 VSS 提供程序。

使用 AWS Tools for Windows PowerShell 安装 VSS 程序包

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 us-east-2 区域。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 执行以下命令下载并安装适用于 Systems Manager 的 VSS 必需组件。

```
Send-SSMCommand -DocumentName AWS-ConfigureAWSPackage -InstanceId "$instance"-Parameter
  @{'action'='Install';'name'='AwsVssComponents'}
```

创建启用 VSS 的 EBS 快照

本部分包括通过使用 Amazon EC2 控制台、AWS CLI 和 Tools for Windows PowerShell 创建启用 VSS 的 EBS 快照的步骤。

主题

- [使用控制台创建启用 VSS 的 EBS 快照 \(p. 51\)](#)
- [使用 AWS CLI 创建启用 VSS 的 EBS 快照 \(p. 52\)](#)
- [使用 AWS Tools for Windows PowerShell 创建启用 VSS 的 EBS 快照 \(p. 53\)](#)

使用控制台创建启用 VSS 的 EBS 快照

使用控制台创建启用 VSS 的 EBS 快照 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择 Run a Command。
4. 在命令文档列表中，选择 AWSEC2-CreateVssSnapshot。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. 在命令参数部分中：
 - a. 从排除启动卷列表选择一个选项。使用此参数从备份过程中排除启动卷。
 - b. 在 Description 字段中，键入描述。此描述适用于此过程 (可选但建议使用) 创建的任何快照。
 - c. 在标签字段中，为要应用于此过程创建的所有快照的标签键入键和值。标签可以帮助您找到、管理和从快照列表还原卷。默认情况下，系统用 Name 键填充标签参数。对于此密钥的值，指定要应用于此过程创建的快照的名称。如果您要指定更多标签，请使用分号分隔标签。例如，
Key=**Environment**, Value=**Test**; Key=**User**, Value=**TestUser1**。

此参数是可选的，但是我们建议您标记快照。默认情况下，系统使用设备 ID 和 AppConsistent 标记快照 (用于表示成功的启用 VSS 的应用程序一致性 EBS 快照)。

7. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
8. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
9. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

10. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

11. 选择 Run。

如果成功，则该命令用新快照填充 EBS 快照列表。您可以通过搜索指定的标签或搜索 AppConsistent 在 EBS 快照列表中查找这些快照。如果命令执行失败，请查看 Systems Manager 命令输出，了解执行失败的详细原因。如果命令成功完成，但特定卷备份失败，您可以在 EBS 卷列表中排查失败的原因。

如果命令失败并且您使用 Systems Manager 和 VPC 终端节点，请确保您配置了 com.amazonaws.**region**.ec2 终端节点。未定义 EC2 终端节点时，枚举所附加 EBS 卷的调用将失败，这会导致 Systems Manager 命令失败。有关使用 Systems Manager 设置 VPC 终端节点的更多信息，请参阅 [设置 Systems Manager 的 VPC 端点 \(p. 13\)](#)。

Note

您可以通过创建使用 AWSEC2-CreateVssSnapshot SSM 文档的维护时段任务自动执行备份。有关更多信息，请参阅 [使用 Maintenance Window \(p. 255\)](#)。

使用 AWS CLI 创建启用 VSS 的 EBS 快照

使用以下过程来利用 AWS CLI 创建启用 VSS 的 EBS 快照。执行此命令时，您可以指定以下参数：

- 实例 (必需)：指定一个或多个 Amazon EC2 Windows 实例。您可以手动指定实例，也可以指定标签。
- 描述 (可选)：指定关于此备份的详细信息。
- 标签 (可选)：指定要分配给快照的键值标签对。标签可以帮助您找到、管理和从快照列表还原卷。默认情况下，系统用 Name 键填充标签参数。对于此密钥的值，指定要应用于此过程创建的快照的名称。您还可以使用以下格式向此列表中添加自定义标签：Key=**Environment**,Value=**Test**;Key=**User**,Value=**TestUser1**。

此参数是可选的，但是我们建议您标记快照。默认情况下，系统使用设备 ID 和 AppConsistent 标记快照 (用于表示成功的启用 VSS 的应用程序一致性 EBS 快照)。

- 排除启动卷 (可选)：使用此参数从备份过程中排除启动卷。

使用 AWS CLI 创建启用 VSS 的 EBS 快照

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
```

```
Default output format [None]: ENTER
```

2. 执行以下命令创建启用 VSS 的 EBS 快照。

```
aws ssm send-command --document-name "AWSEC2-CreateVssSnapshot" --instance-ids "i-12345678" --parameters '{"ExcludeBootVolume":["False"],"description":["Description"],"tags":["Key=key_name,Value=tag_value"]}'
```

如果成功，则该命令用新快照填充 EBS 快照列表。您可以通过搜索指定的标签或搜索 `AppConsistent` 在 EBS 快照列表中查找这些快照。如果命令执行失败，请查看命令输出，了解执行失败的详细原因。

您可以通过创建使用 `AWSEC2-CreateVssSnapshot` SSM 文档的维护时段任务自动执行备份。有关更多信息，请参阅 [使用 Maintenance Window \(p. 255\)](#)。

使用 AWS Tools for Windows PowerShell 创建启用 VSS 的 EBS 快照

使用以下过程来利用 AWS Tools for Windows PowerShell 创建启用 VSS 的 EBS 快照。执行此命令时，您可以指定以下参数：

- 实例 (必需)：指定一个或多个 Amazon EC2 Windows 实例。您可以手动指定实例，也可以指定标签。
- 描述 (可选)：指定关于此备份的详细信息。
- 标签 (可选)：指定要分配给快照的键值标签对。标签可以帮助您找到、管理和从快照列表还原卷。默认情况下，系统用 `Name` 键填充标签参数。对于此密钥的值，指定要应用于此过程创建的快照的名称。您还可以使用以下格式向此列表中添加自定义标签：Key=`Environment`,Value=`Test`;Key=`User`,Value=`TestUser1`。

此参数是可选的，但是我们建议您标记快照。默认情况下，系统使用设备 ID 和 `AppConsistent` 标记快照 (用于表示成功的启用 VSS 的应用程序一致性 EBS 快照)。

- 排除启动卷 (可选)：使用此参数从备份过程中排除启动卷。

使用 AWS Tools for Windows PowerShell 创建启用 VSS 的 EBS 快照

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 `us-east-2` 区域。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 执行以下命令创建启用 VSS 的 EBS 快照。

```
Send-SSMCommand -DocumentName AWSEC2-CreateVssSnapshot -InstanceId "$instance" -Parameter @{'ExcludeBootVolume'='False';'description'='a_description';'tags'='Key=key_name,Value=tag_value'}
```

如果成功，则该命令用新快照填充 EBS 快照列表。您可以通过搜索指定的标签或搜索 `AppConsistent` 在 EBS 快照列表中查找这些快照。如果命令执行失败，请查看命令输出，了解执行失败的详细原因。如果命令成功完成，但特定卷备份失败，您可以在 EBS 快照列表中排查失败的原因。

您可以通过创建使用 AWSEC2-CreateVssSnapshot SSM 文档的维护时段任务自动执行备份。有关更多信息，请参阅 [使用 Maintenance Window \(p. 255\)](#)。

通过使用 AWSEC2-ManageVssIO SSM 文档 (高级) 创建启用 VSS 的 EBS 快照

您可以使用以下脚本和预定义的 AWSEC2-ManageVssIO SSM 文档来临时暂停 I/O，创建启用 VSS 的 EBS 快照，并重新启动 I/O。此过程在执行命令的用户环境中运行。如果用户具有足够的权限来创建和标记快照，那么 Systems Manager 可以创建和标记启用 VSS 的 EBS 快照，而实例上无需更多 IAM 快照角色。

相比之下，AWSEC2-CreateVssSnapshot 文档需要您向要为其创建 EBS 快照的每个实例分配 IAM 快照角色。如果您出于策略或合规性原因不希望向您的实例提供其他 IAM 权限，则可以使用以下脚本。

开始前的准备工作

请注意有关该过程的以下重要详细信息：

- 此过程使用一条 PowerShell 脚本 (CreateVssSnapshotAdvancedScript.ps1)，为您指定的实例上的所有卷 (根卷除外) 拍摄快照。如果您需要为根卷拍摄快照，则必须使用 AWSEC2-CreateVssSnapshot SSM 文档。
- 该脚本会调用 AWSEC2-ManageVssIO 文档两次。第一次将 Action 参数设置为 Freeze，这会暂停实例上的所有 I/O。第二次将 Action 参数设置为 Thaw，这会强制恢复 I/O。
- 请勿在未使用 CreateVssSnapshotAdvancedScript.ps1 脚本的情况下尝试使用 AWSEC2-ManageVssIO 文档。VSS 中的限制要求调用 Freeze 和 Thaw 操作相隔不超过十秒钟，在不使用脚本的情况下手动调用这些操作会导致错误。

通过使用 AWSEC2-ManageVssIO SSM 文档创建启用 VSS 的 EBS 快照

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 us-east-2 区域。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 下载 [CreateVssSnapshotAdvancedScript.zip](#) 文件并提取文件内容。
4. 在简单文本编辑器中打开脚本，并在脚本底部编辑示例调用，然后运行该调用。

如果成功，则该命令用新快照填充 EBS 快照列表。您可以通过搜索指定的标签或搜索 AppConsistent 在 EBS 快照列表中查找这些快照。如果命令执行失败，请查看命令输出，了解执行失败的详细原因。如果命令成功完成，但特定卷备份失败，您可以在 EBS 卷列表中排查失败的原因。

从启用 VSS 的 EBS 快照还原卷

您可以使用 RestoreVssSnapshotSampleScript.ps1 脚本从启用 VSS 的 EBS 快照还原实例上的卷。此脚本执行以下任务：

- 停止实例

- 从实例中删除所有现有驱动器 (如果已排除启动卷, 则启动卷除外)
- 从快照创建新卷
- 通过在快照上使用设备 ID 标签将卷附加到实例
- 重新启动实例

Important

以下脚本将断开附加到实例的所有卷, 然后从快照创建新卷。确保已正确备份实例。不会删除旧卷。如果需要, 您可以编辑脚本来删除旧卷。

从启用 VSS 的 EBS 快照还原卷

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限, 或者您必须在 IAM 中被授予相应权限。有关更多信息, 请参阅 [Systems Manager 先决条件](#) (p. 6)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 us-east-2 区域。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 下载 [RestoreVssSnapshotSampleScript.zip](#) 文件并提取文件内容。
4. 在简单文本编辑器中打开脚本, 并在脚本底部编辑示例调用, 然后运行该调用。

将 Chef InSpec 配置文件与 Systems Manager Compliance 结合使用

Systems Manager 现在与 [Chef InSpec](#) 集成。InSpec 是一个开源的运行时框架, 您可以使用它在 GitHub 或 Amazon S3 上创建人工可读的配置文件。然后, 您可以使用 Systems Manager 运行合规性扫描并查看合规和不合规的实例。配置文件列出您的计算环境的安全性、合规性或策略要求。例如, 您可以使用 Systems Manager Compliance 创建配置文件以便在您扫描实例时执行以下检查:

- 检查特定端口是否处于打开或关闭状态。
- 检查特定应用程序是否正在运行。
- 检查某些程序包是否已安装。
- 检查 Windows 注册表项中的特定属性。

您可以为使用 Systems Manager 管理的 Amazon EC2 实例和本地服务器或虚拟机 (VM) 创建 InSpec 配置文件。以下示例 Chef InSpec 配置文件将检查端口 22 是否处于打开状态。

```
control 'Scan Port' do
  impact 10.0
  title 'Server: Configure the service port'
  desc 'Always specify which port the SSH server should listen to.
  Prevent unexpected settings.'
  describe sshd_config do
    its('Port') { should eq('22') }
  end
end
```

InSpec 包含一组资源，可帮助您快速编写检查和审计控制。InSpec 使用 [InSpec 域特定语言 \(DSL\)](#) 在 Ruby 中编写这些控件。您还可以使用由大型 InSpec 用户社区创建的配置文件。例如，GitHub 上的 [DevSec chef-os-hardening](#) 项目包含许多配置文件，可帮助您保护实例和服务器。您可以在 GitHub 或 Amazon Simple Storage Service (Amazon S3) 中创作和存储配置文件。

如何使用

下面演示了 InSpec 配置文件与 Systems Manager Compliance 结合使用的过程。

1. 要么识别要使用的预定义 InSpec 配置文件，要么创建您自己的配置文件。您可以使用 GitHub 上的 [预定义配置文件](#) 来开始操作。有关如何创建您自己的 InSpec 配置文件的信息，请参阅 [使用 InSpec 实现合规自动化](#)。
2. 将配置文件存储在公共或私有 GitHub 存储库或 Amazon S3 存储桶中。
3. 使用 AWS-RunInSpecChecks SSM 文档对您的 InSpec 配置文件运行合规性检查。您可以使用 Run Command (对于按需扫描) 开始合规性扫描，也可以使用 状态管理器 安排定期合规性扫描。
4. 使用 Compliance API 或 Systems Manager Compliance 控制台识别不合规的实例。

Note

Chef 使用您的实例上的客户端来处理配置文件。您不需要安装客户端。当 Systems Manager 执行 AWS-RunInSpecChecks SSM 文档时，系统会检查客户端是否已安装。如果未安装，Systems Manager 会在扫描期间安装 Chef 客户端，然后在扫描完成后卸载客户端。

运行 InSpec 合规性扫描

本节包含有关如何使用 Systems Manager 控制台和 AWS CLI 执行 InSpec Compliance 扫描的信息。控制台过程显示如何配置状态管理器 来运行扫描。AWS CLI 过程显示如何配置 Run Command 来运行扫描。

使用控制台通过 状态管理器 运行 InSpec 合规性扫描

使用 AWS Systems Manager 控制台通过 状态管理器 运行 InSpec 合规性扫描

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 状态管理器。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 状态管理器。

3. 选择创建关联。
4. 在提供关联详细信息部分，输入一个名称。
5. 在命令文档列表中，选择 AWS-RunInSpecChecks。
6. 在文档版本列表中，选择运行时的最新版本。
7. 在参数部分的源类型列表中，选择 GitHub 或 S3。

如果选择 GitHub，则在源信息字段中输入公有或私有 GitHub 存储库中 InSpec 配置文件的路径。以下是由 Systems Manager 团队从以下位置提供的公有配置文件的示例路径：<https://github.com/aws-labs/amazon-ssm/tree/master/Compliance/InSpec/PortCheck>。

```
{ "owner": "aws-labs", "repository": "amazon-ssm", "path": "Compliance/InSpec/PortCheck", "getOptions": "branch:master" }
```

如果您选择 S3，请在源信息字段中输入 Amazon S3 存储桶中 InSpec 配置文件的有效 URL。

有关 Systems Manager 如何与 GitHub 和 Amazon S3 集成的更多信息，请参阅[从 GitHub 和 Amazon S3 运行脚本 \(p. 38\)](#)。

- 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

- 在指定计划部分，使用计划生成器选项创建您想要运行合规性扫描的时间计划。
- 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

- 选择创建关联。系统将创建关联并自动运行合规性扫描。
- 等待几分钟，以完成扫描，然后在导航窗格中选择合规性。
- 在对应的托管实例中，找到合规性类型列为 Custom:Inspec 的实例。
- 选择一个实例 ID 来查看不合规状态的详细信息。

使用 AWS CLI 通过 Run Command 运行 InSpec 合规性扫描

- 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅[Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

- 执行下列命令之一以从 GitHub 或 Amazon S3 运行 InSpec 配置文件。

命令使用以下参数：

- sourceType : GitHub 或 Amazon S3
- sourceInfo : GitHub 或 Amazon S3 存储桶中 InSpec 配置文件文件夹的 URL。该文件夹必须包含基本 InSpec 文件 (*.yml) 和所有相关控件 (*.rb)。

GitHub

```
aws ssm send-command --document-name "AWS-RunInSpecChecks" --targets
'[{ "Key": "tag:tag_name", "Values": ["tag_value"] }]' --parameters '{"sourceType":
["GitHub"], "sourceInfo": [{"owner": "owner_name", "repository":
"repository_name", "path": "Inspec.yml_file"}]}'
```

以下是示例。


```
aws ssm send-command --document-name "AWS-RunInSpecChecks" --targets
'[{ "Key": "tag:testEnvironment", "Values": ["webServers"] }]' --parameters '{ "sourceType":
["GitHub"], "getOptions": "branch:master", "sourceInfo": [ { "owner": "awslabs",
"repository": "amazon-ssm", "path": "Compliance/InSpec/PortCheck" } ] }'
```

Amazon S3

```
aws ssm send-command --document-name "AWS-RunInSpecChecks" --
targets '[{ "Key": "tag:tag_name", "Values": ["tag_value"] }]' --
parameters '{ "sourceType": ["S3"], "sourceInfo": [ { "path": "https://
s3.amazonaws.com/directory/Inspec.yml_file" } ] }'
```

以下是示例。

```
aws ssm send-command --document-name "AWS-RunInSpecChecks" --targets
'[{ "Key": "tag:testEnvironment", "Values": ["webServers"] }]' --parameters '{ "sourceType":
["S3"], "sourceInfo": [ { "path": "https://s3.amazonaws.com/Compliance/InSpec/
PortCheck.yml" } ] }'
```

3. 执行以下命令以查看合规性扫描的摘要。

```
aws ssm list-resource-compliance-summaries --filters
Key=ComplianceType,Values=Custom:Inspec
```

4. 执行以下命令以深入查看不合规的实例。

```
aws ssm list-compliance-items --resource-ids instance_ID --resource-type
ManagedInstance --filters Key=DocumentName,Values=AWS-RunInSpecChecks
```

相关 AWS 服务

以下相关服务可帮助您评估合规性并使用 Chef。

- [Amazon Inspector](#) 可让您根据中央互联网安全 (CIS) 标准中描述的常见漏洞对您的实例执行安全评估。
- [AWS OpsWorks for Chef Automate](#) 允许您在 AWS 中运行 Chef Automate 服务器。

AWS Systems Manager 资源组

资源组是同一 AWS 区域中与查询中提供的条件相匹配的 AWS 资源的集合。您可以在资源组控制台构建查询，或者将查询作为自变量传递给 AWS CLI 中的资源组命令。借助资源组，您可以创建自定义控制台，根据在标签中指定的条件来组织和整合信息。在资源组中创建组后，可使用自动化等 AWS Systems Manager 工具简化您的资源组上的管理任务。您也可以将组作为在 Systems Manager 中查看监控和配置见解的基础。有关更多信息，请参阅 [AWS 资源组](#)。

AWS Systems Manager 见解

AWS Systems Manager (以前称作 Amazon EC2 Systems Manager) 提供以下特性和功能，用于集中查看有关 AWS 资源的数据。

主题

- [AWS Systems Manager Inventory Manager \(p. 60\)](#)
- [AWS Systems Manager 配置合规性 \(p. 92\)](#)

AWS Systems Manager Inventory Manager

您可以使用 AWS Systems Manager 清单从 Amazon EC2 实例和本地服务器或混合环境中的虚拟机 (VM) 中收集操作系统 (OS)、应用程序和实例元数据。您可以查询元数据以快速了解哪些实例运行的是软件策略需要的软件和配置，以及哪些实例需要更新。

内容

- [开始使用 Inventory \(p. 60\)](#)
- [关于 Systems Manager Inventory \(p. 60\)](#)
- [配置清单收集 \(p. 69\)](#)
- [配置清单的资源数据同步 \(p. 71\)](#)
- [查询清单收集 \(p. 73\)](#)
- [删除自定义清单 \(p. 74\)](#)
- [Systems Manager Inventory Manager 演练 \(p. 82\)](#)
- [排除 Systems Manager Inventory 故障 \(p. 91\)](#)

开始使用 Inventory

要开始使用 Inventory，请完成以下任务。

任务	了解更多信息
验证 Systems Manager 先决条件。	Systems Manager 先决条件 (p. 6)
通过创建 Systems Manager 状态管理器 关联来配置 Inventory。	配置清单收集 (p. 69) (Amazon EC2 控制台) 演练：使用 AWS CLI 收集清单 (p. 84)

关于 Systems Manager Inventory

当您配置 Systems Manager Inventory 时，您可以指定要收集的元数据类型、应从中收集元数据的实例以及元数据收集的计划。这些配置将与您的 AWS 账户一起作为 Systems Manager 状态管理器 关联内容保存。关联就是配置。

Note

Inventory 仅收集元数据。它不收集任何个人数据或专有数据。

主题

- [创建 AWS 账户中所有托管实例的清单 \(p. 61\)](#)
- [清单管理中的可配置数据类型 \(p. 61\)](#)
- [清单收集的元数据 \(p. 62\)](#)
- [使用文件和 Windows 注册表清单 \(p. 66\)](#)
- [使用自定义清单 \(p. 67\)](#)
- [相关 AWS 服务 \(p. 68\)](#)

创建 AWS 账户中所有托管实例的清单

您可以通过创建全局清单关联，轻松创建您的 AWS 账户中所有托管实例的清单。全局清单关联执行以下操作：

- 自动将全局清单配置 (关联) 应用到您的 AWS 账户中的所有现有托管实例。应用和运行全局清单关联时，将跳过已具有清单关联的实例。跳过某个实例时，详细状态消息将说明 *Overridden By Explicit Inventory Association*。全局关联跳过了这些实例，但在它们运行所分配的清单关联时，仍会报告清单。
- 将在您的 AWS 账户中创建的新实例自动添加到全局清单关联。

Note

如果为全局清单关联配置了某个实例，并且您将特定关联分配到该实例，则 Systems Manager 清单会降低全局关联的优先级并应用该特定关联。

在配置清单收集时，您可在 AWS Systems Manager 控制台中选择通过选择 *Selecting all managed instances in this account* (选择此账户中的所有托管实例) 目标选项来配置全局清单关联。有关更多信息，请参阅 [配置清单收集 \(p. 69\)](#)。要使用 AWS CLI 创建全局清单关联，请为 `instanceIds` 值使用通配符选项，如以下示例中所示：

```
aws ssm create-association --name AWS-GatherSoftwareInventory --targets
Key=InstanceIds,Values=* --schedule-expression "rate(1 day)" --parameters
applications=Enabled,awsComponents=Enabled,customInventory=Enabled,instanceDetailedInformation=Enabled
```

Note

全局清单关联在 SSM 代理版本 2.0.790.0 或更高版本中可用。有关如何在实例上更新 SSM 代理的信息，请参阅 [示例：更新 SSM 代理 \(p. 183\)](#)。

清单管理中的可配置数据类型

下表介绍了您要配置的 Inventory 收集的不同方面。

配置	详细信息
收集的信息类型	<p>您可以配置清单以收集以下类型的元数据：</p> <ul style="list-style-type: none">• 应用程序：应用程序名称、发布者和版本等。• AWS 组件：EC2 驱动程序、代理、版本等。

配置	详细信息
	<ul style="list-style-type: none"> 文件：名称、大小、版本、安装日期、修改时间和上次访问时间等。 网络配置：IP 地址、MAC 地址、DNS、网关、子网掩码等。 Windows 更新：修补程序 ID、安装者、安装日期等。 实例详细信息：系统名称、操作系统名称、操作系统版本、最后一次启动、DNS、域、工作组和操作系统架构等。 服务：名称、显示名称、状态、依赖服务、服务类型、启动类型等。 Windows 注册表：注册表项路径、值名称、值类型和值。 Windows 角色：名称、显示名称、路径、功能类型、安装状态等。 自定义清单：按照 使用自定义清单 (p. 67) 中所述分配给托管实例的元数据。 <p>Note</p> <p>要查看清单收集的所有元数据的列表，请参阅 清单收集的元数据 (p. 62)。</p>
从中收集信息的实例	您可以选择创建您 AWS 账户中所有实例的清单，单独选择实例，或者使用 Amazon EC2 标签来确定目标实例组。有关针对所有实例执行清单收集的更多信息，请参阅 创建 AWS 账户中所有托管实例的清单 (p. 61) 。
何时收集信息	您可以指定几分钟、几小时、几天和几周的收集间隔。最短收集间隔为 30 分钟。

根据所收集的数据量，系统可能需要几分钟将数据报告到您指定的输出中。收集信息后，元数据会通过安全 HTTPS 通道发送到一个只能从您的 AWS 账户访问的纯文本 AWS 存储。您可以在指定的 Amazon S3 存储桶中查看数据，或在托管实例的清单选项卡上的 Amazon EC2 控制台上查看数据。Inventory 选项卡带有若干预定义的筛选器，可以帮助您查询数据。

要开始在托管实例上收集清单，请参阅 [配置清单收集 \(p. 69\)](#) 和 [演练：使用 AWS CLI 收集清单 \(p. 84\)](#)。

清单收集的元数据

以下示例显示了一份由每个清单插件收集的元数据完整清单。

```
[
  {
    "typeName": "AWS:InstanceInformation",
    "version": "1.0",
    "attributes": [
      { "name": "AgentType", "dataType": "STRING"},
      { "name": "AgentVersion", "dataType": "STRING"},
      { "name": "ComputerName", "dataType": "STRING"},
      { "name": "IamRole", "dataType": "STRING"}
    ]
  }
]
```

```

        { "name": "InstanceId",                "dataType": "STRING"},
        { "name": "IpAddress",                 "dataType": "STRING"},
        { "name": "PlatformName",              "dataType": "STRING"},
        { "name": "PlatformType",              "dataType": "STRING"},
        { "name": "PlatformVersion",           "dataType": "STRING"},
        { "name": "ResourceType",              "dataType": "STRING"}
    ]
},
{
    "typeName": "AWS:Application",
    "version": "1.1",
    "attributes": [
        { "name": "Name",                      "dataType": "STRING"},
        { "name": "ApplicationType",           "dataType": "STRING"},
        { "name": "Publisher",                 "dataType": "STRING"},
        { "name": "Version",                   "dataType": "STRING"},
        { "name": "InstalledTime",             "dataType": "STRING"},
        { "name": "Architecture",              "dataType": "STRING"},
        { "name": "URL",                       "dataType": "STRING"},
        { "name": "Summary",                   "dataType": "STRING"},
        { "name": "PackageId",                  "dataType": "STRING"},
        { "name": "Release",                    "dataType": "STRING"},
        { "name": "Epoch",                     "dataType": "STRING"}
    ]
},
{
    "typeName": "AWS:File",
    "version": "1.0",
    "attributes": [
        { "name": "Name",                      "dataType": "STRING"},
        { "name": "Size",                      "dataType": "STRING"},
        { "name": "Description",                "dataType": "STRING"},
        { "name": "FileVersion",                "dataType": "STRING"},
        { "name": "InstalledDate",              "dataType": "STRING"},
        { "name": "ModificationTime",           "dataType": "STRING"},
        { "name": "LastAccessTime",             "dataType": "STRING"},
        { "name": "ProductName",                "dataType": "STRING"},
        { "name": "InstalledDir",               "dataType": "STRING"},
        { "name": "ProductLanguage",            "dataType": "STRING"},
        { "name": "CompanyName",                "dataType": "STRING"},
        { "name": "ProductVersion",             "dataType": "STRING"}
    ]
},
{
    "typeName": "AWS:AWSComponent",
    "version": "1.0",
    "attributes": [
        { "name": "Name",                      "dataType": "STRING"},
        { "name": "ApplicationType",           "dataType": "STRING"},
        { "name": "Publisher",                 "dataType": "STRING"},
        { "name": "Version",                   "dataType": "STRING"},
        { "name": "InstalledTime",             "dataType": "STRING"},
        { "name": "Architecture",              "dataType": "STRING"},
        { "name": "URL",                       "dataType": "STRING"}
    ]
},
{
    "typeName": "AWS:WindowsUpdate",
    "version": "1.0",
    "attributes": [
        { "name": "HotFixId",                  "dataType": "STRING"},
        { "name": "Description",                "dataType": "STRING"},
        { "name": "InstalledTime",              "dataType": "STRING"},
        { "name": "InstalledBy",                "dataType": "STRING"}
    ]
}
},

```

```
{
  "typeName": "AWS:Network",
  "version": "1.0",
  "attributes": [
    { "name": "Name", "dataType": "STRING" },
    { "name": "SubnetMask", "dataType": "STRING" },
    { "name": "Gateway", "dataType": "STRING" },
    { "name": "DHCPSTable", "dataType": "STRING" },
    { "name": "DNSSTable", "dataType": "STRING" },
    { "name": "MacAddress", "dataType": "STRING" },
    { "name": "IPV4", "dataType": "STRING" },
    { "name": "IPV6", "dataType": "STRING" }
  ]
},
{
  "typeName": "AWS:PatchSummary",
  "version": "1.0",
  "attributes": [
    { "name": "PatchGroup", "dataType": "STRING" },
    { "name": "BaselineId", "dataType": "STRING" },
    { "name": "SnapshotId", "dataType": "STRING" },
    { "name": "OwnerInformation", "dataType": "STRING" },
    { "name": "InstalledCount", "dataType": "NUMBER" },
    { "name": "InstalledOtherCount", "dataType": "NUMBER" },
    { "name": "NotApplicableCount", "dataType": "NUMBER" },
    { "name": "MissingCount", "dataType": "NUMBER" },
    { "name": "FailedCount", "dataType": "NUMBER" },
    { "name": "OperationType", "dataType": "STRING" },
    { "name": "OperationStartTime", "dataType": "STRING" },
    { "name": "OperationEndTime", "dataType": "STRING" }
  ]
},
{
  "typeName": "AWS:ComplianceItem",
  "version": "1.0",
  "attributes": [
    { "name": "ComplianceType", "dataType": "STRING" },
    { "name": "ExecutionId", "dataType": "STRING" },
    { "name": "ExecutionType", "dataType": "STRING" },
    { "name": "ExecutionTime", "dataType": "STRING" },
    { "name": "Id", "dataType": "STRING" },
    { "name": "Title", "dataType": "STRING" },
    { "name": "Status", "dataType": "STRING" },
    { "name": "Severity", "dataType": "STRING" },
    { "name": "DocumentName", "dataType": "STRING" },
    { "name": "DocumentVersion", "dataType": "STRING" },
    { "name": "Classification", "dataType": "STRING" },
    { "name": "PatchBaselineId", "dataType": "STRING" },
    { "name": "PatchSeverity", "dataType": "STRING" },
    { "name": "PatchState", "dataType": "STRING" },
    { "name": "PatchGroup", "dataType": "STRING" },
    { "name": "InstalledTime", "dataType": "STRING" }
  ]
},
{
  "typeName": "AWS:ComplianceSummary",
  "version": "1.0",
  "attributes": [
    { "name": "ComplianceType", "dataType": "STRING" },
    { "name": "PatchGroup", "dataType": "STRING" },
    { "name": "PatchBaselineId", "dataType": "STRING" },
    { "name": "Status", "dataType": "STRING" },
    { "name": "OverallSeverity", "dataType": "STRING" },
    { "name": "ExecutionId", "dataType": "STRING" },
    { "name": "ExecutionType", "dataType": "STRING" },
    { "name": "ExecutionTime", "dataType": "STRING" }
  ]
}
```

```

    { "name": "CompliantCriticalCount",          "dataType": "NUMBER"},
    { "name": "CompliantHighCount",             "dataType": "NUMBER"},
    { "name": "CompliantMediumCount",           "dataType": "NUMBER"},
    { "name": "CompliantLowCount",              "dataType": "NUMBER"},
    { "name": "CompliantInformationalCount",    "dataType": "NUMBER"},
    { "name": "CompliantUnspecifiedCount",      "dataType": "NUMBER"},
    { "name": "NonCompliantCriticalCount",      "dataType": "NUMBER"},
    { "name": "NonCompliantHighCount",          "dataType": "NUMBER"},
    { "name": "NonCompliantMediumCount",        "dataType": "NUMBER"},
    { "name": "NonCompliantLowCount",           "dataType": "NUMBER"},
    { "name": "NonCompliantInformationalCount", "dataType": "NUMBER"},
    { "name": "NonCompliantUnspecifiedCount",   "dataType": "NUMBER"}
  ]
},
{
  "typeName": "AWS:InstanceDetailedInformation",
  "version": "1.0",
  "attributes": [
    { "name": "CPUModel",          "dataType": "STRING"},
    { "name": "CPUCores",         "dataType": "NUMBER"},
    { "name": "CPUs",              "dataType": "NUMBER"},
    { "name": "CPUSpeedMHz",       "dataType": "NUMBER"},
    { "name": "CPUSockets",        "dataType": "NUMBER"},
    { "name": "CPUHyperThreadEnabled", "dataType": "STRING"},
    { "name": "OSServicePack",      "dataType": "STRING"}
  ]
},
{
  "typeName": "AWS:Service",
  "version": "1.0",
  "attributes": [
    { "name": "Name",              "dataType": "STRING"},
    { "name": "DisplayName",       "dataType": "STRING"},
    { "name": "ServiceType",       "dataType": "STRING"},
    { "name": "Status",            "dataType": "STRING"},
    { "name": "DependentServices", "dataType": "STRING"},
    { "name": "ServicesDependedOn", "dataType": "STRING"},
    { "name": "StartType",         "dataType": "STRING"}
  ]
},
{
  "typeName": "AWS:WindowsRegistry",
  "version": "1.0",
  "attributes": [
    { "name": "KeyPath",          "dataType": "STRING"},
    { "name": "ValueName",        "dataType": "STRING"},
    { "name": "ValueType",        "dataType": "STRING"},
    { "name": "Value",            "dataType": "STRING"}
  ]
},
{
  "typeName": "AWS:WindowsRole",
  "version": "1.0",
  "attributes": [
    { "name": "Name",              "dataType": "STRING"},
    { "name": "DisplayName",       "dataType": "STRING"},
    { "name": "Path",              "dataType": "STRING"},
    { "name": "FeatureType",       "dataType": "STRING"},
    { "name": "DependsOn",         "dataType": "STRING"},
    { "name": "Description",       "dataType": "STRING"},
    { "name": "Installed",         "dataType": "STRING"},
    { "name": "InstalledState",    "dataType": "STRING"},
    { "name": "SubFeatures",       "dataType": "STRING"},
    { "name": "ServerComponentDescriptor", "dataType": "STRING"},
    { "name": "Parent",            "dataType": "STRING"}
  ]
}
]

```

```
}  
]
```

Note

随着 2.5 版本的发布，RPM 包管理器将 Serial 属性替换成了 Epoch。与 Serial 一样，Epoch 属性是一个单调递增整数。当您使用 AWS:Application 类型创建清单时，请注意，Epoch 的值越大，表示版本越新。如果 Epoch 值相同或为空，请使用 Version 或 Release 属性值来确定较新版本。

使用文件和 Windows 注册表清单

借助 Systems Manager 清单可以搜索和清点 Windows 和 Linux 操作系统上的文件。您还可以搜索并清点 Windows 注册表。

文件：您可以收集关于文件的元数据信息，包括文件名称、文件创建时间、文件上次修改和访问时间以及文件大小等等。要开始收集文件清单，您需要指定要执行清单的文件路径、定义需要清点的文件类型的一个或多个模式，以及是否应以递归的方式遍历路径。Systems Manager 会清点与模式匹配的指定路径中的文件的所有文件元数据。清单文件使用以下参数输入。

```
{  
  "Path": string,  
  "Pattern": array[string],  
  "Recursive": true,  
  "DirScanLimit" : number // Optional  
}
```

- 路径：您要清点文件的目录路径。对于 Windows，您可以使用 %PROGRAMFILES% 等环境变量，前提是该变量要映射到单个目录路径。例如，如果您使用映射到多个目录路径的 %PATH%，则清单会引发错误。
- 模式：确定文件的一组模式。
- 递归：指示清单是否应以递归方式遍历目录的布尔值。
- DirScanLimit：指定要扫描多少目录的可选值。使用此参数可以将对实例的性能影响降至最低。默认情况下，清单扫描最多 5000 个目录。

Note

清单在所有指定路径中收集最多 500 个文件的元数据。

下面是一些在执行文件清单时如何指定参数的示例。

- 在 Linux 上，会收集 /home/ec2-user 目录 (不包括所有子目录) 中的 .sh 文件中的元数据。

```
[{"Path":"/home/ec2-user","Pattern":["*.sh", "*.sh"],"Recursive":false}]
```

- 在 Windows 上，会以递归方式收集程序文件夹 (包括子目录) 中的所有“.exe”文件的元数据。

```
[{"Path":"C:\\Program Files","Pattern":["*.exe"],"Recursive":true}]
```

- 在 Windows 上，会收集指定日志模式的元数据。

```
[{"Path":"C:\\ProgramData\\Amazon","Pattern":["*amazon*.log"],"Recursive":true}]
```

- 在执行递归集合时会限制目录计数。

```
[{"Path":"C:\\Users","Pattern":["*.ps1"],"Recursive":true, "DirScanLimit": 1000}]
```

Windows 注册表：您可以收集 Windows 注册表项和值。您可以选择一个键路径并以递归方式收集所有键和值。您还可以收集特定路径的特定注册表项及其值。清单会收集键路径、名称、类型和值。

```
{
  "Path": string,
  "Recursive": boolean,
  "ValueNames": array[string] // optional
}
```

- 路径：注册表项的路径。
- 递归：指示清单是否应以递归方式遍历注册表路径的布尔值。
- ValueNames：执行注册表项的清单的一组值名称。如果使用此参数，Systems Manager 仅将清点指定路径的指定值名称。

Note

清单针对所有指定路径收集最多 250 个注册表项值。

下面是一些在执行 Windows 注册表清单时如何指定参数的示例。

- 以递归方式针对特定路径收集所有键和值。

```
[{"Path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\Amazon", "Recursive": true}]
```

- 针对特定路径收集所有键和值 (禁用递归搜索)。

```
[{"Path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\Intel\\PSIS\\PSIS_DECODER", "Recursive": false}]
```

- 使用 ValueNames 选项收集特定键。

```
{"Path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\Amazon\\MachineImage", "ValueNames": ["AMIName"]}
```

使用自定义清单

您可以通过创建自定义清单将所需的任何元数据分配给您的实例。例如，假如您负责管理数据中心多个机架中的大量服务器，而且这些服务器已配置为 Systems Manager 托管实例。目前，您在电子表格中存储服务器机架位置的相关信息。借助自定义清单，您可以指定每个实例的机架位置作为实例上的元数据。当您使用 Systems Manager 收集清单时，该元数据与其他清单元数据一起收集。随后您可以使用[资源数据同步](#)将所有清单元数据传输到中央 Amazon S3 存储桶并查询数据。

要将自定义清单分配给实例，可以使用 Systems Manager [PutInventory](#) API 操作，如 [演练：将自定义清单元数据分配给某个实例 \(p. 82\)](#) 中所述。或者，您可以创建自定义清单 JSON 文件并将其上传到实例。本部分描述了如何创建 JSON 文件。

```
{
  "SchemaVersion": "1.0",
  "TypeName": "Custom:RackInformation",
  "Content": {
    "Location": "US-EAST-02.CMH.RACK1",
    "InstalledTime": "2016-01-01T01:01:01Z",
    "vendor": "DELL",
    "Zone": "BJS12",
    "TimeZone": "UTC-8"
  }
}
```


如下例所示，您在文件中还可以指定多个项目。

```
{
  "SchemaVersion": "1.0",
  "TypeName": "Custom:PuppetModuleInfo",
  "Content": [{
    "Name": "puppetlabs/aws",
    "Version": "1.0"
  },
  {
    "Name": "puppetlabs/dsc",
    "Version": "2.0"
  }
]
```

自定义清单的 JSON 架构需要 SchemaVersion、TypeName 和 Content 部分，但您可以定义这些部分的信息。

```
{
  "SchemaVersion": "user_defined",
  "TypeName": "Custom:user_defined",
  "Content": {
    "user_defined_attribute1": "user_defined_value1",
    "user_defined_attribute2": "user_defined_value2",
    "user_defined_attribute3": "user_defined_value3",
    "user_defined_attribute4": "user_defined_value4"
  }
}
```

TypeName 最长为 100 个字符。此外，TypeName 部分必须以 Custom 开头。例如，Custom:PuppetModuleInfo。Custom 和您指定的##必须以一个大写字母开头。以下示例可能会引起异常：“CUSTOM:RackInformation”、“custom:rackinformation”。

Content 部分包括属性和##。这些项目不区分大小写。但是，如果您定义了属性（例如“Vendor”: “DELL”），在自定义清单文件中必须一致地引用此属性。如果您在一个文件中指定“Vendor”: “DELL”（vendor 使用大写字母“V”），在另一个文件中指定“vendor”: “DELL”（vendor 使用小写字母“v”），系统会返回错误。

Note

您必须使用 .json 扩展名保存文件。

创建文件之后，您必须在实例上保存文件。下表说明了自定义清单 JSON 文件必须存储在实例中的哪个位置：

操作系统	路径
Windows	%SystemDrive%\ProgramData\Amazon\SSM\InstanceData\<instance-id>\inventory\custom
Linux	/var/lib/amazon/ssm/<instance-id>/inventory/custom

有关如何使用自定义清单的示例，请参阅[使用 EC2 Systems Manager 自定义清单类型获取队列的磁盘利用率](#)。

相关 AWS 服务

Systems Manager Inventory 可以提供您当前清单的快照，以便帮助您管理软件策略并改善整个队列的安全状况。您可以使用以下 AWS 服务来扩展清单管理和迁移功能。

- AWS Config 可以提供清单的更改历史记录，并支持创建规则，用于在配置项更改时生成通知。有关更多信息，请参阅 AWS Config Developer Guide 中的[记录 Amazon EC2 托管实例清单](#)。
- AWS Application Discovery Service 旨在从您的本地 VM 中收集有关操作系统类型、应用程序清单、流程、连接和服务器性能指标的清单，以便支持成功迁移到 AWS。有关更多信息，请参阅[Application Discovery Service User Guide](#)。

配置清单收集

本部分介绍了如何使用 Amazon EC2 控制台在一个或多个托管实例上配置清单收集。本部分还介绍如何使用 Systems Manager 资源数据同步在单个 Amazon S3 存储桶中聚合多个 AWS 账户和区域的清单数据。有关如何使用 AWS CLI 来配置清单收集的示例，请参阅[Systems Manager Inventory Manager 演练 \(p. 82\)](#)。

开始前的准备工作

配置清单收集前，请完成以下任务。

- 更新需要清点的实例上的 SSM 代理。通过运行最新版本的 SSM 代理，可确保您能够收集所有支持的清单类型的元数据。有关如何使用 状态管理器 更新 SSM 代理 的信息，请参阅[演练：自动更新 SSM 代理 \(CLI\) \(p. 286\)](#)。
- 验证您的实例是否满足 Systems Manager 先决条件。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。
- (可选) 创建用于收集自定义清单的 JSON 文件。有关更多信息，请参阅 [使用自定义清单 \(p. 67\)](#)。

配置收集

请按照以下过程使用控制台配置对托管实例的清单收集。

Note

配置清单收集时，首先创建 Systems Manager 状态管理器 关联。Systems Manager 会在关联运行时收集清单数据。如果您不先创建关联，并试图使用 Run Command 等调用 aws:softwareInventory 插件，则系统会返回以下错误：

```
The aws:softwareInventory plugin can only be invoked via ssm-associate.
```

还要注意，一个实例一次只能配置一个 Inventory 关联。如果您为实例配置了两个或更多关联，则关联不会运行，而且系统不会收集清单数据。

配置清单收集

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Inventory。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Inventory。

3. 选择设置清单。
4. 在目标部分中，通过选择以下选项之一来标识您要运行此操作的实例。
 - Selecting all managed instances in this account (选择此账户中的所有托管实例) – 此选项选择没有现有清单关联的所有托管实例。如果您选择此选项，则在清单收集期间将跳过所有已有清单关联的实例，然后在清单结果中显示状态为已跳过。有关更多信息，请参阅 [创建 AWS 账户中所有托管实例的清单 \(p. 61\)](#)。

- 指定标签 – 此选项可让您指定单个标签，用于标识您的账户中要收集其清单的实例。如果使用标签，则将来使用相同标签创建的所有实例也将报告清单。如果现有清单关联到所有实例，可以针对 All managed instances (所有托管实例) 目标组中的不同清单覆盖实例成员资格，使用标签选择特定实例作为目标。未来通过 All managed instances (所有托管实例) 选项进行清单收集的时候，将跳过具有指定标签的实例。
 - 手动选择实例 – 此选项可让您选择账户中的特定托管实例。使用此选项明确选择特定实例将覆盖 All managed instances (所有托管实例) 目标上的清单关联。未来通过 All managed instances (所有托管实例) 进行清单收集的时候，将跳过该实例。
5. 在计划部分，选择您希望系统从实例中收集清单元数据的频率。
 6. 在 Parameters 部分，使用列表启用或禁用不同类型的清单收集。如果要针对文件或 Windows 注册表创建清单搜索，请参阅以下示例。

文件

- 在 Linux 上，会收集 /home/ec2-user 目录 (不包括所有子目录) 中的 .sh 文件中的元数据。

```
[{"Path":"/home/ec2-user","Pattern":["*.sh","*.sh"],"Recursive":false}]
```

- 在 Windows 上，会以递归方式收集程序文件夹 (包括子目录) 中的所有“.exe”文件的元数据。

```
[{"Path":"C:\\Program Files","Pattern":["*.exe"],"Recursive":true}]
```

- 在 Windows 上，会收集指定日志模式的元数据。

```
[{"Path":"C:\\ProgramData\\Amazon","Pattern":["*amazon*.log"],"Recursive":true}]
```

- 在执行递归集合时会限制目录计数。

```
[{"Path":"C:\\Users","Pattern":["*.ps1"],"Recursive":true, "DirScanLimit": 1000}]
```

Windows 注册表

- 以递归方式针对特定路径收集所有键和值。

```
[{"Path":"HKEY_LOCAL_MACHINE\\SOFTWARE\\Amazon","Recursive": true}]
```

- 针对特定路径收集所有键和值 (禁用递归搜索)。

```
[{"Path":"HKEY_LOCAL_MACHINE\\SOFTWARE\\Intel\\PSIS\\PSIS_DECODER", "Recursive": false}]
```

- 使用 ValueNames 选项收集特定键。

```
{"Path":"HKEY_LOCAL_MACHINE\\SOFTWARE\\Amazon\\MachineImage","ValueNames":["AMIName"]}
```

有关收集文件和 Windows 注册表清单的更多信息，请参阅[使用文件和 Windows 注册表清单 \(p. 66\)](#)。

7. 如果需要在 Amazon S3 存储桶中存储关联执行状态，请在 Advanced 部分中选择 Sync inventory execution logs to an S3 bucket。
8. 选择设置清单。Systems Manager 创建一个 状态管理器 关联并立即在实例上运行清单。
9. 在导航窗格中，选择 状态管理器。验证是否使用 AWS-GatherSoftwareInventory 文档创建了新的关联。此外，确认 Status 字段显示 Success。如果选择 Sync inventory execution logs to an S3 bucket 选项，

则过几分钟后，可以在 Amazon S3 中查看日志数据。要查看特定实例的清单数据，请在导航窗格中选择 Managed Instances。

10. 选择一个实例，然后选择 View details。

11. 在实例详细信息页面上，选择 Inventory。使用 Inventory type 列表筛选清单。

在配置清单收集后，我们建议您配置 Systems Manager 资源数据同步。资源数据同步将所有清单数据集中在目标 Amazon S3 存储桶中，当收集新的清单数据时，将会自动更新中央存储。所有清单数据存储在目标 Amazon S3 存储桶中后，您可以使用 Amazon Athena 和 Amazon QuickSight 等服务查询和分析聚合数据。有关更多信息，请参阅 [配置清单的资源数据同步](#) (p. 71)。

配置清单的资源数据同步

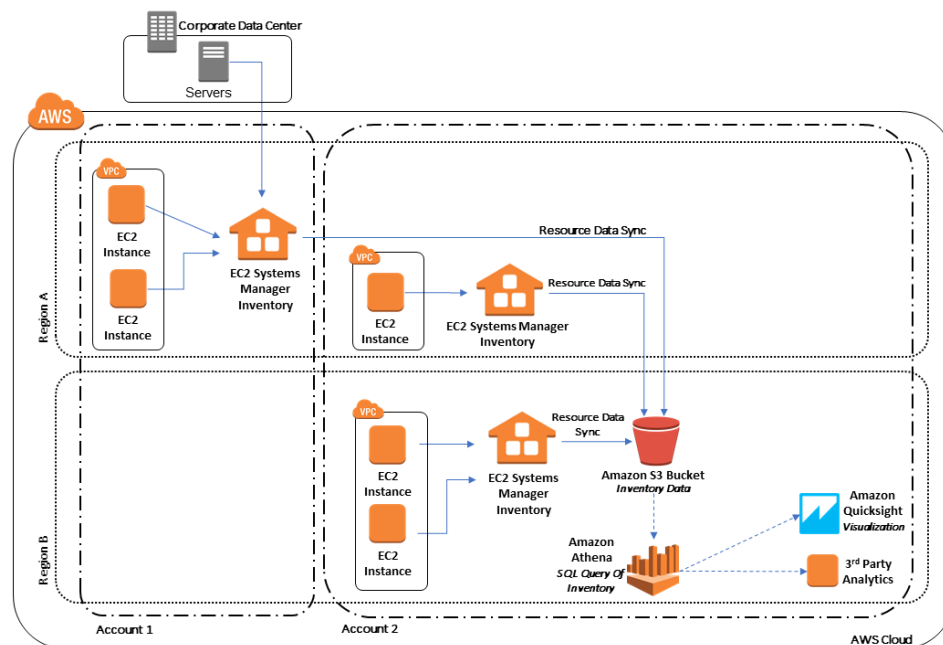
您可以使用 Systems Manager 资源数据同步将所有托管实例收集的清单数据存储到单个 Amazon S3 存储桶。收集新的清单数据后，资源数据同步自动更新集中式数据。所有清单数据存储在目标 Amazon S3 存储桶中后，您可以使用 Amazon Athena 和 Amazon QuickSight 等服务查询和分析聚合数据。

例如，假设您将清单配置为收集关于操作系统 (OS) 和 150 个托管实例机群上运行的应用程序的数据。其中部分实例位于混合数据中心，而其他实例在多个 AWS 区域中的 Amazon EC2 中运行。如果您未配置清单的资源数据同步，则需要手动收集已收集的每个实例的清单数据，或者必须创建脚本来收集这些信息。然后您需要将数据传输到应用程序中，以便运行查询和分析数据。

使用资源数据同步，您可以通过一次性操作同步所有托管实例中的所有清单数据。在创建同步时，可以指定多个 AWS 账户和 AWS 区域的托管实例。在成功创建同步后，Systems Manager 创建所有清单数据的基准，并将其保存在目标 Amazon S3 存储桶中。当收集新的清单数据时，Systems Manager 自动更新 Amazon S3 存储桶中的数据。然后您可以快速有效地将数据传输到 Amazon Athena 和 Amazon QuickSight。

图 1 显示了资源数据同步如何将 Amazon EC2 和混合环境中的托管实例的清单数据聚合到目标 Amazon S3 存储桶。该图还显示了资源数据同步如何处理多个 AWS 账户和 AWS 区域。

图 1：对多个 AWS 账户和 AWS 区域进行资源数据同步



如果删除托管实例，资源数据同步会保留删除的实例的清单文件。但是，对于正在运行的实例，当新文件创建并写入到 Amazon S3 存储桶中时，资源数据同步自动覆盖旧清单文件。如果要随着时间推移跟踪清单

变化，您可以使用 AWS Config 服务跟踪 ManagedInstanceInventory 资源类型。有关更多信息，请参见 [AWS Config 入门](#)。

相关内容

- 资源数据同步使用下列 API 操作：[CreateResourceDataSync](#)、[ListResourceDataSync](#) 和 [DeleteResourceDataSync](#)。
- Amazon QuickSight 是一项业务分析服务，可轻松构建可视化内容，以便您分析数据并从数据中获得见解。单击一次即可从 QuickSight 连接到 Athena。您不需要提供终端节点或用户名和密码。您可以选择 Athena 作为您的数据源，选择要分析的数据库和表，然后开始在 QuickSight 中实现数据可视化。有关更多信息，请参阅 [Amazon QuickSight User Guide](#)。
- Amazon Athena 是一种交互式查询服务，可使用此服务通过标准 SQL 查询在 Amazon S3 中轻松分析数据。Athena 无需运行 Amazon EC2 实例，因此无需管理基础设施。您只需为您运行的查询付费。有关更多信息，请参阅 [Amazon Athena User Guide](#)。

主题

- [创建清单的资源数据同步 \(p. 72\)](#)

创建清单的资源数据同步

通过 Amazon S3 和 AWS Systems Manager 控制台，使用以下过程创建清单的资源数据同步。您也可以使用 AWS CloudFormation 创建或删除资源数据同步。要使用 AWS CloudFormation，请将 [AWS::SSM::ResourceDataSync](#) 资源添加到您的 AWS CloudFormation 模板。相关信息请参阅 AWS CloudFormation User Guide 中的 [使用 AWS CloudFormation 模板](#)。

Note

您可以使用 AWS Key Management Service (AWS KMS) 加密 Amazon S3 存储桶中的清单数据。有关如何使用 AWS CLI 创建加密同步和如何处理 Amazon Athena 和 Amazon QuickSight 中的集中式数据的示例，请参阅 [演练：使用资源数据同步聚合清单数据 \(p. 86\)](#)。

创建和配置一个 Amazon S3 存储桶用于资源数据同步

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. 创建用来存储聚合清单数据的存储桶。有关更多信息，请参阅 Amazon Simple Storage Service Getting Started Guide 中的 [创建存储桶](#)。请记住存储桶名称和创建此存储桶的 AWS 区域。
3. 选择 Permissions 选项卡，然后选择 Bucket Policy。
4. 将下面的存储桶策略复制并粘贴到策略编辑器中。使用您创建的 Amazon S3 存储桶的名称和有效的 AWS 账户 ID 替换 *bucket-name* 和 *account-id*。或者，使用 Amazon S3 前缀 (子目录) 的名称替换 *bucket-prefix*。如果您未创建前缀，则从以下策略的 ARN 中删除 *bucket-prefix*。

Note

有关查看您的 AWS 账户 ID 的信息，请参阅 IAM User Guide 中的 [AWS 账户 ID 及其别名](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SSMBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::bucket-name"
    }
  ]
}
```

```
{
  "Sid": " SSMBucketDelivery",
  "Effect": "Allow",
  "Principal": {
    "Service": "ssm.amazonaws.com"
  },
  "Action": "s3:PutObject",
  "Resource": ["arn:aws:s3::bucket-name/bucket-prefix/*/accountid=account-
id/*"],
  "Condition": {
    "StringEquals": {
      "s3:x-amz-acl": "bucket-owner-full-control"
    }
  }
}
```

创建资源数据同步

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择托管实例。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Managed Instances。

3. 选择 Resource Data Syncs，然后选择 Create resource data sync。
4. 在 Sync name 字段中，键入同步配置的名称。
5. 在 Bucket name 字段中，键入在此程序开始时创建的 Amazon S3 存储桶的名称。
6. (可选) 在 Bucket prefix 字段中，键入 Amazon S3 存储桶前缀 (子目录) 的名称。
7. 在 Bucket region 字段中，如果创建的 Amazon S3 存储桶位于当前 AWS 区域，则选择 This region。如果存储桶位于不同的 AWS 区域，则选择 Another region，然后键入该区域的名称。

Note

如果同步和目标 Amazon S3 存储桶位于不同区域，您可能需要支付数据传输价格。有关更多信息，请参阅 [Amazon S3 定价](#)。

8. 在 KMS 密钥 ARN 字段中，键入或粘贴 KMS 密钥 ARN 来加密 Amazon S3 中的清单数据。
9. 选择 Create。

查询清单收集

收集清单数据后，可以使用 Systems Manager 中的筛选功能查询满足特定筛选条件的托管实例列表。

基于清单筛选条件查询实例 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Inventory。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Inventory。

3. 在 Filter by resource groups, tags or inventory types 部分中，选择筛选框。此时会显示预定义筛选器列表。

4. 选择要筛选的属性。例如，选择 AWS:Application。如果出现提示，选择要筛选的二级属性。例如，选择 AWS:Application.Name。
5. 从列表中选择分隔符。例如，选择 Begin with。此时筛选器中会显示一个文本框。
6. 在文本框中键入一个值。例如，键入 Amazon (SSM 代理命名为 Amazon SSM Agent)。
7. 按 Enter。系统返回一个托管实例列表，其中包含以 Amazon 开头的应用程序名称。

Note

您可以组合多个筛选条件来细化搜索。

删除自定义清单

您可以使用 [DeleteInventory](#) API 操作来删除自定义清单类型以及与该类型关联的数据。您可以使用 AWS CLI 调用 `delete-inventory` 命令来删除某个清单类型的所有数据。您可以使用 `SchemaDeleteOption` 调用 `delete-inventory` 命令来删除自定义清单类型。

Note

清单类型也称为清单架构。

`SchemaDeleteOption` 参数包括以下选项：

- `DeleteSchema`：此选项删除指定的自定义类型和与之关联的所有数据。如果需要，可以稍后重新创建架构。
- `DisableSchema`：如果您选择此选项，系统将禁用当前版本，删除其所有数据，并在版本早于或等于所禁用版本时忽略所有新数据。您可以通过对高于已禁用版本的版本调用 [PutInventory](#) 操作来重新启用此清单类型。

使用 AWS CLI 删除或禁用自定义清单

1. 在您的本地计算机上，[下载](#) 最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令来使用 `dry-run` 选项查看将从系统中删除哪些数据。此命令不会删除任何数据。

```
aws ssm delete-inventory --type-name "Custom:custom_type_name" --dry-run
```

系统将返回类似于以下内容的信息。

```
{
  "DeletionSummary":{
    "RemainingCount":3,
    "SummaryItems":[
      {
        "Count":2,
```

```

        "RemainingCount":2,
        "Version":"1.0"
      },
      {
        "Count":1,
        "RemainingCount":1,
        "Version":"2.0"
      }
    ],
    "TotalCount":3
  },
  "TypeName":"Custom:custom_type_name"
}

```

有关如何了解和删除清单摘要的信息，请参阅[了解删除清单摘要 \(p. 78\)](#)。

4. 执行以下命令来删除自定义清单类型的所有数据。

```
aws ssm delete-inventory --type-name "Custom:custom_type_name"
```

Note

此命令的输出未显示删除进度。因此，TotalCount 和 Remaining Count 始终相同，因为系统尚未删除任何数据。您可以使用 describe-inventory-deletions 命令来显示删除进度，如本主题的下文所述。

系统将返回类似于以下内容的信息。

```

{
  "DeletionId":"system_generated_deletion_ID",
  "DeletionSummary":{
    "RemainingCount":3,
    "SummaryItems":[
      {
        "Count":2,
        "RemainingCount":2,
        "Version":"1.0"
      },
      {
        "Count":1,
        "RemainingCount":1,
        "Version":"2.0"
      }
    ],
    "TotalCount":3
  },
  "TypeName":"custom_type_name"
}

```

系统将从 Systems Manager Inventory 服务中删除指定的自定义清单类型的所有数据。

5. 执行下面的命令。此命令对清单类型的当前版本执行以下操作：禁用当前版本、删除它的所有数据并忽略所有新数据 (如果版本低于或等于已禁用版本)。

```
aws ssm delete-inventory --type-name "Custom:custom_type_name" --schema-delete-option "DisableSchema"
```

系统将返回类似于以下内容的信息。

```

{
  "DeletionId":"system_generated_deletion_ID",

```



```
"DeletionSummary":{
  "RemainingCount":3,
  "SummaryItems":[
    {
      "Count":2,
      "RemainingCount":2,
      "Version":"1.0"
    },
    {
      "Count":1,
      "RemainingCount":1,
      "Version":"2.0"
    }
  ],
  "TotalCount":3
},
"TypeName":"Custom:custom_type_name"
}
```

您可以使用以下命令查看已禁用的清单类型。

```
aws ssm get-inventory-schema --type-name Custom:custom_type_name
```

6. 执行以下命令来删除清单类型。

```
aws ssm delete-inventory --type-name "Custom:custom_type_name" --schema-delete-option
"DeleteSchema"
```

系统将删除指定的自定义类型的架构和所有清单数据。

系统将返回类似于以下内容的信息。

```
{
  "DeletionId":"system_generated_deletion_ID",
  "DeletionSummary":{
    "RemainingCount":3,
    "SummaryItems":[
      {
        "Count":2,
        "RemainingCount":2,
        "Version":"1.0"
      },
      {
        "Count":1,
        "RemainingCount":1,
        "Version":"2.0"
      }
    ],
    "TotalCount":3
  },
  "TypeName":"Custom:custom_type_name"
}
```

查看删除状态

您可以使用 `describe-inventory-deletions` AWS CLI 命令来查看删除操作的状态。您可以指定删除 ID 来查看特定删除操作的状态。或者，您也可以省略删除 ID 来查看最近 30 天执行的所有删除的列表。

1. 执行以下命令来查看删除操作的状态。系统将返回 `delete-inventory` 摘要中的删除 ID。

```
aws ssm describe-inventory-deletions --deletion-id system_generated_deletion_ID
```

系统将返回最新状态。删除操作可能尚未完成。系统将返回类似于以下内容的信息。

```
{ "InventoryDeletions":
  [
    { "DeletionId": "system_generated_deletion_ID",
      "DeletionStartTime": 1521744844,
      "DeletionSummary":
        { "RemainingCount": 1,
          "SummaryItems":
            [
              { "Count": 1,
                "RemainingCount": 1,
                "Version": "1.0" }
            ],
          "TotalCount": 1 },
      "LastStatus": "InProgress",
      "LastStatusMessage": "The Delete is in progress",
      "LastStatusUpdateTime": 1521744844,
      "TypeName": "Custom:custom_type_name" }
  ]
}
```

如果删除操作成功，则 LastStatusMessage 指出：Deletion is successful。

```
{ "InventoryDeletions":
  [
    { "DeletionId": "system_generated_deletion_ID",
      "DeletionStartTime": 1521744844,
      "DeletionSummary":
        { "RemainingCount": 0,
          "SummaryItems":
            [
              { "Count": 1,
                "RemainingCount": 0,
                "Version": "1.0" }
            ],
          "TotalCount": 1 },
      "LastStatus": "Complete",
      "LastStatusMessage": "Deletion is successful",
      "LastStatusUpdateTime": 1521745253,
      "TypeName": "Custom:custom_type_name" }
  ]
}
```

2. 执行以下命令来查看最近 30 天执行的所有删除的列表。

```
aws ssm describe-inventory-deletions --max-results a number
```

```
{ "InventoryDeletions":
  [
    { "DeletionId": "system_generated_deletion_ID",
      "DeletionStartTime": 1521682552,
      "DeletionSummary":
        { "RemainingCount": 0,
          "SummaryItems":
            [
              { "Count": 1,
```

```
        "RemainingCount": 0,
        "Version": "1.0"}
    ],
    "TotalCount": 1},
    "LastStatus": "Complete",
    "LastStatusMessage": "Deletion is successful",
    "LastStatusUpdateTime": 1521682852,
    "TypeName": "Custom:custom_type_name",
    {"DeletionId": "system_generated_deletion_ID",
    "DeletionStartTime": 1521744844,
    "DeletionSummary":
    {"RemainingCount": 0,
    "SummaryItems":
    [
        {"Count": 1,
        "RemainingCount": 0,
        "Version": "1.0"}
    ]
    },
    "TotalCount": 1},
    "LastStatus": "Complete",
    "LastStatusMessage": "Deletion is successful",
    "LastStatusUpdateTime": 1521745253,
    "TypeName": "Custom:custom_type_name",
    {"DeletionId": "system_generated_deletion_ID",
    "DeletionStartTime": 1521680145,
    "DeletionSummary":
    {"RemainingCount": 0,
    "SummaryItems":
    [
        {"Count": 1,
        "RemainingCount": 0,
        "Version": "1.0"}
    ]
    },
    "TotalCount": 1},
    "LastStatus": "Complete",
    "LastStatusMessage": "Deletion is successful",
    "LastStatusUpdateTime": 1521680471,
    "TypeName": "Custom:custom_type_name"}
    ],
    "NextToken": "next-token"
```

了解删除清单摘要

为帮助您了解删除清单摘要的内容，请考虑以下示例。用户将 Custom:RackSpace 清单分配给了三个实例。清单项目 1 和 2 使用自定义类型版本 1.0 ("SchemaVersion":"1.0")。清单项目 3 使用自定义类型版本 2.0 ("SchemaVersion":"2.0")。

RackSpace 自定义清单 1

```
{
  "CaptureTime":"2018-02-19T10:48:55Z",
  "TypeName":"CustomType:RackSpace",
  "InstanceId":"i-1234567890",
  "SchemaVersion":"1.0"   "Content":[
    {
      content of custom type omitted
    }
  ]
}
```

RackSpace 自定义清单 2

```
{
  "CaptureTime":"2018-02-19T10:48:55Z",
  "TypeName":"CustomType:RackSpace",
  "InstanceId":"i-1234567891",
  "SchemaVersion":"1.0"   "Content":[
    {
      content of custom type omitted
    }
  ]
}
```

RackSpace 自定义清单 3

```
{
  "CaptureTime":"2018-02-19T10:48:55Z",
  "TypeName":"CustomType:RackSpace",
  "InstanceId":"i-1234567892",
  "SchemaVersion":"2.0"   "Content":[
    {
      content of custom type omitted
    }
  ]
}
```

用户运行以下命令来预览将删除哪些数据。

```
aws ssm delete-inventory --type-name "Custom:RackSpace" --dry-run
```

系统将返回类似于以下内容的信息。

```
{
  "DeletionId":"1111-2222-333-444-66666",
  "DeletionSummary":{
    "RemainingCount":3,
    "TotalCount":3,
    TotalCount and RemainingCount are the number of items that would be deleted
    if this was not a dry run. These numbers are the same because the system didn't delete
    anything.
    "SummaryItems":[
      {
        "Count":2,
        The system found two items that use SchemaVersion 1.0.
        Neither item was deleted.
        "RemainingCount":2,
        "Version":"1.0"
      },
      {
        "Count":1,
        The system found one item that uses SchemaVersion 1.0.
        This item was not deleted.
        "RemainingCount":1,
        "Version":"2.0"
      }
    ],
  },
  "TypeName":"Custom:RackSpace"
}
```

用户运行以下命令来删除 Custom:RackSpace 清单。

Note

此命令的输出未显示删除进度。因此，TotalCount 和 Remaining Count 始终相同，因为系统尚未删除任何数据。您可以使用 describe-inventory-deletions 命令来显示删除进度。

```
aws ssm delete-inventory --type-name "Custom:RackSpace"
```

系统将返回类似于以下内容的信息。

```
{
  "DeletionId": "1111-2222-333-444-7777777",
  "DeletionSummary": {
    "RemainingCount": 3,
    "SummaryItems": [
      {
        "Count": 2,
        "RemainingCount": 2,
        "Version": "1.0"
      },
      {
        "Count": 1,
        "RemainingCount": 1,
        "Version": "2.0"
      }
    ],
    "TotalCount": 3
  },
  "TypeName": "RackSpace"
}
```

在 CloudWatch Events 中查看清单删除操作

您可以配置 Amazon CloudWatch Events，在用户删除自定义清单时创建事件。CloudWatch 为自定义清单删除操作提供三种类型的事件：

- 实例的删除操作：特定托管实例的自定义清单是否已成功删除。
- 删除操作摘要：删除操作的摘要。
- 针对禁用的自定义清单类型的警告：如果用户对之前禁用的自定义清单类型版本调用了 [PutInventory](#) API 操作，则会发出警告事件。

下面是每个事件的示例：

实例的删除操作

```
{
  "version": "0",
  "id": "998c9cde-56c0-b38b-707f-0411b3ff9d11",
  "detail-type": "Inventory Resource State Change",
  "source": "aws.ssm",
  "account": "478678815555",
  "time": "2018-05-24T22:24:34Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:478678815555:managed-instance/i-0a5feb270fc3f0b97"
  ],
  "detail": {
    "action-status": "succeeded",
    "action": "delete",
    "resource-type": "managed-instance",
    "resource-id": "i-0a5feb270fc3f0b97",
  }
}
```

```
    "action-reason": "",
    "type-name": "Custom:MyInfo"
  }
}
```

删除操作摘要

```
{
  "version": "0",
  "id": "83898300-f576-5181-7a67-fb3e45e4fad4",
  "detail-type": "Inventory Resource State Change",
  "source": "aws.ssm",
  "account": "478678815555",
  "time": "2018-05-24T22:28:25Z",
  "region": "us-east-1",
  "resources": [
  ],
  "detail": {
    "action-status": "succeeded",
    "action": "delete-summary",
    "resource-type": "managed-instance",
    "resource-id": "",
    "action-reason": "The delete for type name Custom:MyInfo was completed. The deletion summary is: {\"totalCount\":2,\"remainingCount\":0,\"summaryItems\":[{\"version\":\"1.0\", \"count\":2,\"remainingCount\":0}]}",
    "type-name": "Custom:MyInfo"
  }
}
```

针对禁用的自定义清单类型的警告

```
{
  "version": "0",
  "id": "49c1855c-9c57-b5d7-8518-b64aeeef5e4a",
  "detail-type": "Inventory Resource State Change",
  "source": "aws.ssm",
  "account": "478678815555",
  "time": "2018-05-24T22:46:58Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ssm:us-east-1:478678815555:managed-instance/i-0ee2d86a2cfc371f6"
  ],
  "detail": {
    "action-status": "failed",
    "action": "put",
    "resource-type": "managed-instance",
    "resource-id": "i-0ee2d86a2cfc371f6",
    "action-reason": "The inventory item with type name Custom:MyInfo was sent with a disabled schema version 1.0. You must send a version greater than 1.0",
    "type-name": "Custom:MyInfo"
  }
}
```

要为自定义清单删除操作创建 CloudWatch Events 规则，请按照以下过程操作。此过程向您介绍如何创建向 Amazon SNS 主题发送自定义清单删除操作通知的规则。开始前，检查您是否拥有 Amazon SNS 主题或者新建一个。有关更多信息，请参见 Amazon Simple Notification Service Developer Guide 中的 [入门](#) 部分。

为删除清单操作配置 CloudWatch Events

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.

2. 在左侧导航窗格中，选择 Events，然后选择 Create rule。
3. 在 Event Source 下，验证已选中 Event Pattern。
4. 在 Service Name 字段中，选择 EC2 Simple Systems Manager (SSM)。
5. 在 Event Type (事件类型) 字段中，选择 Inventory (清单)。
6. 检查是否已选择 Any detail type (任何详细信息类型)，然后选择 Add targets (添加目标)。
7. 在 Select target type (选择目标类型) 列表中，选择 SNS topic (SNS 主题)，然后从列表中选择您的主题。
8. 在 Configure input (配置输入) 列表中，检查 Matched event (已匹配事件) 是否处于选中状态。
9. 选择 Configure details。
10. 指定名称和描述，然后选择 Create rule (创建规则)。

Systems Manager Inventory Manager 演练

使用以下演练收集和管理清单数据。我们建议您首先在测试环境中使用托管实例执行这些演练。

开始前的准备工作

在开始这些演练之前，请完成下列任务。

- 更新需要清点的实例上的 SSM 代理。通过运行最新版本的 SSM 代理，可确保您能够收集所有支持的清单类型的元数据。有关如何使用 状态管理器 更新 SSM 代理 的信息，请参阅[演练：自动更新 SSM 代理 \(CLI\)](#) (p. 286)。
- 验证您的实例是否满足 Systems Manager 先决条件。有关更多信息，请参阅[Systems Manager 先决条件](#) (p. 6)。
- (可选) 创建用于收集自定义清单的 JSON 文件。有关更多信息，请参阅[使用自定义清单](#) (p. 67)。

内容

- [演练：将自定义清单元数据分配给某个实例](#) (p. 82)
- [演练：使用 AWS CLI 收集清单](#) (p. 84)
- [演练：使用资源数据同步聚合清单数据](#) (p. 86)

演练：将自定义清单元数据分配给某个实例

以下步骤为您演示了使用 `PutInventory` API 操作将自定义清单元数据分配给托管实例的过程。此示例将机架位置信息分配给某个实例。有关自定义清单的更多信息，请参阅[使用自定义清单](#) (p. 67)。

将自定义清单元数据分配给某个实例

1. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令，以便将机架位置信息分配给某个实例。

```
aws ssm put-inventory --instance-id ID --items '[{"CaptureTime":  
  "2016-08-22T10:01:01Z", "TypeName": "Custom:RackInfo", "Content":[{"RackLocation":  
    "Bay B/Row C/Rack D/Shelf E"}], "SchemaVersion": "1.0"}]'
```

4. 执行以下命令以便查看该实例的自定义清单条目。

```
aws ssm list-inventory-entries --instance-id ID --type-name "Custom:RackInfo"
```

系统会使用类似以下形式的信息进行响应。

```
{  
  "InstanceId": "ID",  
  "TypeName": "Custom:RackInfo",  
  "Entries": [  
    {  
      "RackLocation": "Bay B/Row C/Rack D/Shelf E"  
    }  
  ],  
  "SchemaVersion": "1.0",  
  "CaptureTime": "2016-08-22T10:01:01Z"  
}
```

5. 执行以下命令以便查看自定义元数据。

```
aws ssm get-inventory
```

系统会使用类似以下形式的信息进行响应。

```
{  
  "Entities": [  
    {  
      "Data": {  
        "AWS:InstanceInformation": {  
          "Content": [  
            {  
              "ComputerName": "WIN-9JHCEPEGORG.WORKGROUP",  
              "InstanceId": "ID",  
              "ResourceType": "EC2Instance",  
              "AgentVersion": "3.19.1153",  
              "PlatformVersion": "6.3.9600",  
              "PlatformName": "Windows Server 2012 R2 Standard",  
              "PlatformType": "Windows"  
            }  
          ],  
          "TypeName": "AWS:InstanceInformation",  
          "SchemaVersion": "1.0"  
        }  
      },  
      "Id": "ID"  
    }  
  ]  
}
```


演练：使用 AWS CLI 收集清单

以下步骤为您演示了使用 Inventory 从 Amazon EC2 实例中收集元数据的过程。配置清单收集时，首先创建 Systems Manager 状态管理器 关联。Systems Manager 会在关联运行时收集清单数据。如果您不先创建关联，并试图使用 Run Command 等调用 aws:softwareInventory 插件，则系统会返回以下错误：

The aws:softwareInventory plugin can only be invoked via ssm-associate.

Note

一个实例一次只能配置一个清单关联。如果您为实例配置了两个或更多清单关联，则关联不会运行，而且系统不会收集清单数据。

从实例收集清单

1. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令，创建一个在实例上运行 Inventory 的状态管理器 关联。此命令将服务配置为每六小时运行一次，并收集实例上的网络配置、Windows 更新和应用程序元数据。

```
aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets
"Key=instanceids,Values=an instance ID" --schedule-expression "cron(0 0/30 * 1/1
* ? *)" --output-location "{ \"S3Location\": { \"OutputS3Region\": \"region-id\",
\"OutputS3BucketName\": \"Test bucket\", \"OutputS3KeyPrefix\": \"Test\" } }" --
parameters "networkConfig=Enabled,windowsUpdates=Enabled,applications=Enabled"
```

region-id 表示实例所在的 AWS 区域，例如，US East (Ohio) Region 的 us-east-2。

系统会使用类似以下形式的信息进行响应。

```
{
  "AssociationDescription": {
    "ScheduleExpression": "cron(0 0/30 * 1/1 * ? *)",
    "OutputLocation": {
      "S3Location": {
        "OutputS3KeyPrefix": "Test",
        "OutputS3BucketName": "Test bucket",
        "OutputS3Region": "us-east-2"
      }
    },
    "Name": "The name you specified",
    "Parameters": {
      "applications": [
        "Enabled"
      ],
      "networkConfig": [
        "Enabled"
      ]
    }
  }
}
```

```

    ],
    "windowsUpdates": [
      "Enabled"
    ]
  },
  "Overview": {
    "Status": "Pending",
    "DetailedStatus": "Creating"
  },
  "AssociationId": "1a2b3c4d5e6f7g-1a2b3c-1a2b3c-1a2b3c-1a2b3c4d5e6f7g",
  "DocumentVersion": "$DEFAULT",
  "LastUpdateAssociationDate": 1480544990.06,
  "Date": 1480544990.06,
  "Targets": [
    {
      "Values": [
        "i-1a2b3c4d5e6f7g"
      ],
      "Key": "InstanceIds"
    }
  ]
}

```

您可以使用 `Targets` 参数来确定带有 EC2 标签的大型目标实例组。例如：

```

aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets
  "Key=tag:Environment,Values=Production" --schedule-expression "cron(0 0/30 * 1/1
  * ? *)" --output-location "{ \"S3Location\": { \"OutputS3Region\": \"us-east-2\",
  \"OutputS3BucketName\": \"Test bucket\", \"OutputS3KeyPrefix\": \"Test\" } }" --
  parameters "networkConfig=Enabled,windowsUpdates=Enabled,applications=Enabled"

```

您还可以使用带有表达式的 `files` 和 `windowsRegistry` 清单类型来清点 Windows 实例上的文件和 Windows 注册表项。有关这些清单类型的更多信息，请参阅[使用文件和 Windows 注册表清单](#) (p. 66)。

```

aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets
  "Key=instanceids,Values=i-0704358e3a3da9eb1" --schedule-expression "cron(0
  0/30 * 1/1 * ? *)" --parameters '{"files":["[\"Path\": \"C:\\Program Files\\",
  \"Pattern\": [\"*.exe\"], \"Recursive\": true}]", "windowsRegistry": ["[\"Path\":
  \"HKEY_LOCAL_MACHINE\\Software\\Amazon\\\", \"Recursive\":true}]]}' --profile dev-pdx

```

4. 执行以下命令查看关联状态。

```

aws ssm describe-instance-associations-status --instance-id an instance ID

```

系统会使用类似以下形式的信息进行响应。

```

{
  "InstanceAssociationStatusInfos": [
    {
      "Status": "Pending",
      "DetailedStatus": "Associated",
      "Name": "reInvent2016PolicyDocumentTest",
      "InstanceId": "i-1a2b3c4d5e6f7g",
      "AssociationId": "1a2b3c4d5e6f7g-1a2b3c-1a2b3c-1a2b3c-1a2b3c4d5e6f7g",
      "DocumentVersion": "1"
    }
  ]
}

```

演练：使用资源数据同步聚合清单数据

以下演练介绍如何使用 AWS CLI 创建资源数据同步配置。资源数据同步自动将所有托管实例的清单数据传输出到中央 Amazon S3 存储桶。当发现新的清单数据时，同步自动更新中央 Amazon S3 存储桶中的数据。此演练还介绍如何使用 Amazon Athena 和 Amazon QuickSight 查询和分析聚合数据。有关使用 Amazon EC2 控制台创建资源数据同步的信息，请参阅 [配置清单的资源数据同步 \(p. 71\)](#)。

Note

本演练包括有关如何使用 AWS Key Management Service (AWS KMS) 加密同步的信息。清单不收集任何用户特定、专有或敏感数据，因此加密是可选的。有关 AWS KMS 的更多信息，请参阅 [AWS Key Management Service Developer Guide](#)。

开始前的准备工作

在开始本演练之前，您必须从您的托管实例收集清单元数据。对于本演练中的 Amazon Athena 和 Amazon QuickSight，建议您收集应用程序元数据。有关如何收集清单数据的更多信息，请参阅 [演练：使用 AWS CLI 收集清单 \(p. 84\)](#)。

(可选) 如果要使用 AWS KMS 加密同步，则必须创建包括以下策略的新密钥，或更新现有密钥并向其添加此策略。

```
{
  "Version": "2012-10-17",
  "Id": "ssm-access-policy",
  "Statement": [
    {
      "Sid": "ssm-access-policy-statement",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Resource": "arn:aws:kms:region:AWS-account-ID:key/KMS-key-id"
    }
  ]
}
```

创建清单的资源数据同步

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. 创建用来存储聚合清单数据的存储桶。有关更多信息，请参阅 Amazon Simple Storage Service Getting Started Guide 中的 [创建存储桶](#)。请记住存储桶名称和创建此存储桶的 AWS 区域。
3. 创建存储桶后，选择 Permissions 选项卡，然后选择 Bucket Policy。
4. 将下面的存储桶策略复制并粘贴到策略编辑器中。使用您创建的 Amazon S3 存储桶的名称和有效的 AWS 账户 ID 替换 *bucket-name* 和 *account-id*。或者，使用 Amazon S3 前缀 (子目录) 的名称替换 *bucket-prefix*。如果您未创建前缀，则从该策略的 ARN 中删除 *bucket-prefix*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SSMBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Resource": "arn:aws:s3:::bucket-name/*"
    }
  ]
}
```

```

        "Action": "s3:GetBucketAcl",
        "Resource": "arn:aws:s3:::bucket-name"
    },
    {
        "Sid": "SSMBucketDelivery",
        "Effect": "Allow",
        "Principal": {
            "Service": "ssm.amazonaws.com"
        },
        "Action": "s3:PutObject",
        "Resource": ["arn:aws:s3:::bucket-name/bucket-prefix/*/accountid=account-
id/*"],
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
        }
    }
]
}

```

5. (可选) 如果要加密同步，则必须将以下策略添加到存储桶。重复上一步骤以向存储桶添加以下策略。

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::bucket-name/prefix/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms",
                    "s3:x-amz-server-side-encryption-aws-kms-key-
id": "arn:aws:kms:region:AWS-account-ID:key/KMS-key-ID"
                }
            }
        }
    ]
}

```

6. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
7. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```

AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER

```

8. (可选) 如果要加密同步，请执行以下命令以验证存储桶策略是否强制执行 KMS 密钥要求。

```
aws s3 cp ./A file in the bucket s3://bucket-name/prefix/ --sse aws:kms --sse-kms-key-
id "arn:aws:kms:region:AWS-account-ID:key/KMS-key-id" --region region
```

9. 执行以下命令，以使用您在此程序开始时创建的 Amazon S3 存储桶创建资源数据同步配置。此命令从您当前已登录的 AWS 区域创建同步。

Note

如果同步和目标 Amazon S3 存储桶位于不同区域，您可能需要支付数据传输价格。有关更多信息，请参阅 [Amazon S3 定价](#)。

```
aws ssm create-resource-data-sync --sync-name a name --s3-destination
"BucketName=the name of the S3 bucket,Prefix=the name of the prefix, if
specified,SyncFormat=JsonSerDe,Region=the region where the S3 bucket was created"
```

您可以使用 `region` 参数指定创建同步配置的位置。在下例中，来自 `us-west-1` 区域的清单数据将同步到 `us-west-2` 区域中的 Amazon S3 存储桶。

```
aws ssm create-resource-data-sync --sync-name InventoryDataWest --s3-destination
"BucketName=InventoryData,Prefix=HybridEnv,SyncFormat=JsonSerDe,Region=us-west-2" --
region us-west-1
```

(可选) 如果要使用 AWS KMS 加密同步，请执行以下命令创建同步。如果您加密同步，则 AWS KMS 密钥和 Amazon S3 存储桶必须位于相同区域。

```
aws ssm create-resource-data-sync --sync-name sync-name --s3-destination
"BucketName=bucket-
name,Prefix=prefix,SyncFormat=JsonSerDe,AWSKMSKeyARN=arn:aws:kms:region:AWS-account-
ID:key/KMS-key-id,Region=bucket-region" --region region
```

10. 执行以下命令查看同步配置的状态。

```
aws ssm list-resource-data-sync
```

如果您在其他区域中创建了同步配置，则必须指定 `region` 参数，如下例所示。

```
aws ssm list-resource-data-sync --region us-west-1
```

11. 在成功创建同步配置后，浏览 Amazon S3 中的目标存储桶。清单数据应在几分钟内显示。

在 Amazon Athena 中处理数据

以下部分介绍如何在 Amazon Athena 中查看和查询数据。在开始之前，我们建议您首先了解 Athena。有关更多信息，请参阅[什么是 Amazon Athena ?](#) 和[使用数据](#) (Amazon Athena User Guide)。

在 Amazon Athena 中查看和查询数据

1. Open the Athena console at <https://console.aws.amazon.com/athena/>.
2. 将以下语句复制并粘贴到查询编辑器中，然后选择 Run Query。

```
CREATE DATABASE ssminventory
```

系统将创建一个名为 `ssminventory` 的数据库。

3. 将以下语句复制并粘贴到查询编辑器中，然后选择 Run Query。使用 Amazon S3 目标的名称和前缀替换 `bucket-name` 和 `bucket-prefix`。

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_Application (
  Name string,
  ApplicationType string,
```

```
Publisher string,
Version string,
InstalledTime string,
Architecture string,
URL string,
Summary string,
PackageId string
)
PARTITIONED BY (AccountId string, Region string, ResourceType string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1'
) LOCATION 's3://bucket-name/bucket-prefix/AWS:Application/'
```

4. 将以下语句复制并粘贴到查询编辑器中，然后选择 Run Query。

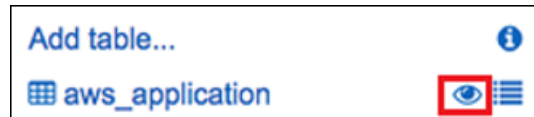
```
MSCK REPAIR TABLE ssminventory.AWS_Application
```

系统将对表进行分区。

Note

如果您从其他 AWS 区域或账户创建资源数据同步，则必须再次运行该命令，以更新分区。您可能还需要更新您的 Amazon S3 存储桶策略。

5. 要预览数据，请选择 AWS_Application 表旁边的视图图标。



6. 将以下语句复制并粘贴到查询编辑器中，然后选择 Run Query。

```
SELECT a.name, a.version, count( a.version) frequency
from aws_application a where
a.name = 'aws-cfn-bootstrap'
group by a.name, a.version
order by frequency desc
```

此查询返回不同版本的 aws-cfn-bootstrap 的计数，这是 Amazon EC2 Linux 和 Windows 实例上出现的 AWS 应用程序。

7. 将以下语句分别复制并粘贴到查询编辑器中，将 *bucket-name* 和 *bucket-prefix* 替换为有关 Amazon S3 的信息，然后选择运行查询。这些语句在 Athena 中设置其他清单表。

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_AWSComponent (
  `ResourceId` string,
  `Name` string,
  `ApplicationType` string,
  `Publisher` string,
  `Version` string,
  `InstalledTime` string,
  `Architecture` string,
  `URL` string
)
PARTITIONED BY (AccountId string, Region string, ResourceType string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1'
) LOCATION 's3://bucket-name/bucket-prefix/AWS:AWSComponent/'

MSCK REPAIR TABLE ssminventory.AWS_AWSComponent
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_WindowsUpdate (  
  `ResourceId` string,  
  `HotFixId` string,  
  `Description` string,  
  `InstalledTime` string,  
  `InstalledBy` string  
)  
PARTITIONED BY (AccountId string, Region string, ResourceType string)  
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  
WITH SERDEPROPERTIES (  
  'serialization.format' = '1'  
) LOCATION 's3://bucket-name/bucket-prefix/AWS:WindowsUpdate/'  
  
MSCK REPAIR TABLE ssminventory.AWS_WindowsUpdate
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_InstanceInformation (  
  `AgentType` string,  
  `AgentVersion` string,  
  `ComputerName` string,  
  `IamRole` string,  
  `InstanceId` string,  
  `IpAddress` string,  
  `PlatformName` string,  
  `PlatformType` string,  
  `PlatformVersion` string  
)  
PARTITIONED BY (AccountId string, Region string, ResourceType string)  
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  
WITH SERDEPROPERTIES (  
  'serialization.format' = '1'  
) LOCATION 's3://bucket-name/bucket-prefix/AWS:InstanceInformation/'  
  
MSCK REPAIR TABLE ssminventory.AWS_InstanceInformation
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_Network (  
  `ResourceId` string,  
  `Name` string,  
  `SubnetMask` string,  
  `Gateway` string,  
  `DHCPServer` string,  
  `DNSServer` string,  
  `MacAddress` string,  
  `IPV4` string,  
  `IPV6` string  
)  
PARTITIONED BY (AccountId string, Region string, ResourceType string)  
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  
WITH SERDEPROPERTIES (  
  'serialization.format' = '1'  
) LOCATION 's3://bucket-name/bucket-prefix/AWS:Network/'  
  
MSCK REPAIR TABLE ssminventory.AWS_Network
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS ssminventory.AWS_PatchSummary (  
  `ResourceId` string,  
  `PatchGroup` string,  
  `BaselineId` string,  
  `SnapshotId` string,  
  `OwnerInformation` string,
```

```
`InstalledCount` int,
`InstalledOtherCount` int,
`NotApplicableCount` int,
`MissingCount` int,
`FailedCount` int,
`OperationType` string,
`OperationStartTime` string,
`OperationEndTime` string
)
PARTITIONED BY (AccountId string, Region string, ResourceType string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = '1'
) LOCATION 's3://bucket-name/bucket-prefix/AWS:PatchSummary/'

MSCK REPAIR TABLE ssminventory.AWS_PatchSummary
```

在 Amazon QuickSight 中处理数据

以下部分提供概述，包含用于在 Amazon QuickSight 中构建可视化内容的链接。

在 Amazon QuickSight 中构建可视化内容

1. 注册 [Amazon QuickSight](#)，然后登录 QuickSight 控制台。
2. 从 AWS_Application 表和您创建的任何其他表创建数据集。有关更多信息，请参阅[使用 Amazon Athena 数据创建数据集](#)。
3. 联接表。例如，您可以联接 AWS_InstanceInformation 中的 instanceid 列，因为它与其他清单表中的 resourceid 列匹配。有关联接表的更多信息，请参阅[联接表](#)。
4. 构建可视化内容。有关更多信息，请参阅[处理 Amazon QuickSight 视觉对象](#)。

排除 Systems Manager Inventory 故障

本主题包括有关如何排除 Systems Manager Inventory 中常见错误或问题的信息。

不支持的代理

如果清单关联的详细状态显示 UnsupportedAgent (不支持的代理)，并且关联状态显示失败，则实例上的 SSM 代理版本不正确。例如，要创建全局清单关联 (为 AWS 账户中的所有实例创建清单)，您需要使用 SSM 代理 版本 2.0.790.0 或更高版本。您可以在托管实例页面的代理版本列中查看各个实例上运行的代理版本。有关如何在实例上更新 SSM 代理的信息，请参阅[示例：更新 SSM 代理 \(p. 183\)](#)。

Skipped

如果某个实例的清单关联状态显示为已跳过，这意味着您创建了全局清单关联，但跳过的实例已有分配的清单关联。全局清单关联未分配到此实例，并且全局清单关联未收集清单。但是，在特定清单关联运行时，实例仍将报告清单数据。

已失败

如果某个实例的清单关联状态显示失败，则意味着实例上分配了多个清单关联。一个实例一次只能分配一个清单关联。清单关联使用 AWS-GatherSoftwareInventory SSM 文档。您可以使用 AWS CLI 执行以下命令来查看实例的关联列表。

```
aws ssm describe-instance-associations-status --instance-id instance ID
```


AWS Systems Manager 配置合规性

您可以使用 AWS Systems Manager 配置合规性扫描托管实例队列，了解补丁合规性和配置不一致性。您可以从多个 AWS 账户和区域中收集并聚合数据，然后深入了解不合规的特定资源。默认情况下，配置合规性将显示有关 Systems Manager Patch Manager 修补和 Systems Manager 状态管理器 关联的合规性数据。您也可以根据 IT 或业务要求自定义服务并创建自己的合规性类型。您可以使用 Systems Manager Run Command、状态管理器 或 Amazon CloudWatch Events 快速修复问题。您也可以将数据传送到 Amazon Athena 和 Amazon QuickSight 来生成队列范围的报告。

配置合规性不另外收取费用。您仅需为实际使用的 AWS 资源付费。

Note

Systems Manager 现在与 [Chef InSpec](#) 集成。InSpec 是一个开源的运行时框架，您可以使用它在 GitHub 或 Amazon S3 上创建人工可读的配置文件。然后，您可以使用 Systems Manager 运行合规性扫描并查看合规和不合规的实例。有关更多信息，请参阅 [将 Chef InSpec 配置文件与 Systems Manager Compliance 结合使用 \(p. 55\)](#)。

内容

- [配置合规性入门 \(p. 92\)](#)
- [关于配置合规性 \(p. 94\)](#)
- [修复合规性问题 \(p. 96\)](#)
- [Systems Manager 配置合规性管理器演练 \(AWS CLI\) \(p. 97\)](#)

配置合规性入门

要开始使用配置合规性，请完成以下任务。

任务	了解更多信息
配置合规性在 Systems Manager 托管实例上使用 Patch Manager 补丁数据、状态管理器 关联和自定义合规类型。通过验证 Systems Manager 先决条件验证您的 Amazon EC2 实例和混合计算机是否配置为托管实例。	Systems Manager 先决条件 (p. 6)
将托管实例上的 SSM 代理更新到最新版本。	安装和配置 SSM 代理 (p. 14)
如果您计划监控补丁合规性，请验证您是否已配置 Systems Manager Patch Manager。您必须使用 Patch Manager 执行修补操作，然后配置合规性才能显示补丁合规性数据。	AWS Systems Manager Patch Manager (p. 204)
如果您计划监控关联合规性，请验证您是否已创建状态管理器 关联。您必须创建关联，配置合规性才能显示关联合规性数据。	AWS Systems Manager 状态管理器 (p. 277)
(可选) 创建自定义合规性类型。	Systems Manager 配置合规性管理器演练 (AWS CLI) (p. 97)
(可选) 创建资源数据同步以将所有合规性数据聚合在一个目标 Amazon S3 存储桶。	为配置合规性创建资源数据同步 (p. 93)

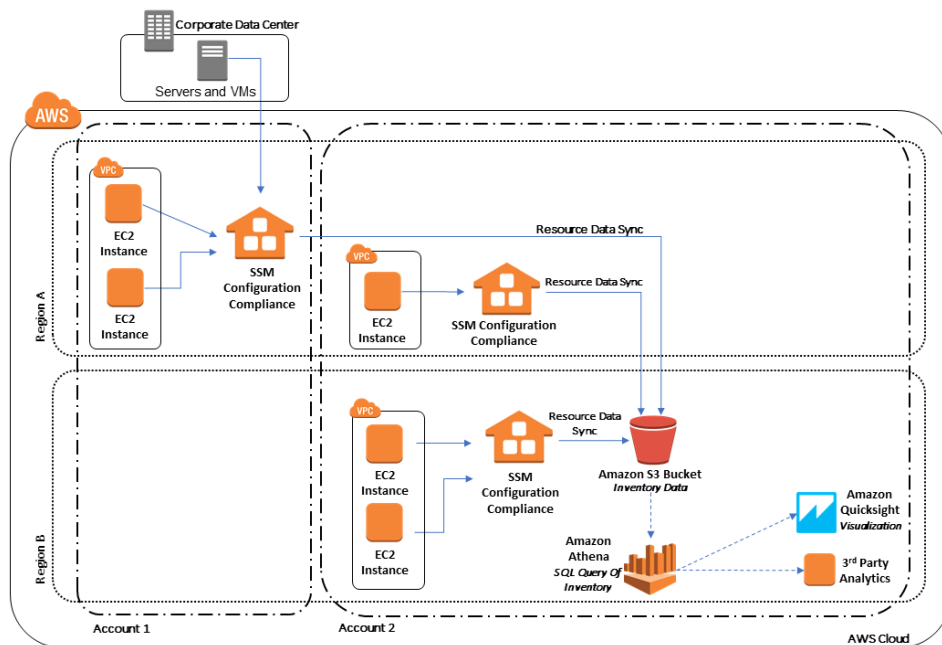
主题

- [为配置合规性创建资源数据同步 \(p. 93\)](#)

为配置合规性创建资源数据同步

您可以使用 Systems Manager 资源数据同步将来自所有托管实例的合规性数据发送到目标 Amazon S3 存储桶。在创建同步时，可以指定多个 AWS 账户、AWS 区域和您的本地混合环境的托管实例。收集新的合规性数据后，资源数据同步自动更新集中式数据。所有合规性数据存储存储在目标 Amazon S3 存储桶中后，您可以使用 Amazon Athena 和 Amazon QuickSight 等服务查询和分析聚合数据。为配置合规性配置资源数据同步是一次性操作。

下图显示资源数据同步如何将来自不同账户、区域和您的混合环境的所有数据聚合到一个中央存储库。



通过使用 Amazon EC2 控制台，使用以下程序创建配置合规性的资源数据同步。

创建和配置一个 Amazon S3 存储桶用于资源数据同步

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. 创建用来存储聚合清单数据的存储桶。有关更多信息，请参阅 Amazon Simple Storage Service Getting Started Guide 中的 [创建存储桶](#)。请记住存储桶名称和创建此存储桶的 AWS 区域。
3. 选择 Permissions 选项卡，然后选择 Bucket Policy。
4. 将下面的存储桶策略复制并粘贴到策略编辑器中。使用您创建的 Amazon S3 存储桶的名称和有效的 AWS 账户 ID 替换 **Bucket-Name** 和 **Account-ID**。或者，使用 Amazon S3 前缀 (子目录) 的名称替换 **Bucket-Prefix**。如果您未创建前缀，则从该策略的 ARN 中删除 **Bucket-Prefix**。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SSMBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::Bucket-Name"
    }
  ]
}
```

```
        "Sid": " SSMBucketDelivery",
        "Effect": "Allow",
        "Principal": {
            "Service": "ssm.amazonaws.com"
        },
        "Action": "s3:PutObject",
        "Resource": [ "arn:aws:s3:::Bucket-Name/Bucket-Prefix/*/"
accountid=Account_ID_number/*"],
        "Condition": {
            "StringEquals": {
                "s3:x-amz-acl": "bucket-owner-full-control"
            }
        }
    }
}
]
```

创建资源数据同步 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择托管实例。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Managed Instances。

3. 选择 Resource Data Syncs，然后选择 Create resource data sync。
4. 在 Sync name 字段中，键入同步配置的名称。
5. 在 Bucket name 字段中，键入在此程序开始时创建的 Amazon S3 存储桶的名称。
6. (可选) 在 Bucket prefix 字段中，键入 Amazon S3 存储桶前缀 (子目录) 的名称。
7. 在 Bucket region 字段中，如果创建的 Amazon S3 存储桶位于当前 AWS 区域，则选择 This region。如果存储桶位于不同的 AWS 区域，则选择 Another region，然后键入该区域的名称。

Note

如果同步和目标 Amazon S3 存储桶位于不同区域，您可能需要支付数据传输价格。有关更多信息，请参阅 [Amazon S3 定价](#)。

8. 选择 Create。

关于配置合规性

本部分包括有关不同类型的信息 (合规性类型) 的信息，您可以使用配置合规性进行查看。配置合规性目前支持 Patch Manager 修补数据、状态管理器 关联和自定义合规性类型。

主题

- [关于实例合规性 \(p. 94\)](#)
- [关于补丁合规性 \(p. 95\)](#)
- [关于关联合规性 \(p. 96\)](#)
- [关于自定义合规性 \(p. 96\)](#)

关于实例合规性

Systems Manager 现在与 [Chef InSpec](#) 集成。InSpec 是一个开源的运行时框架，您可以使用它在 GitHub 或 Amazon S3 上创建人工可读的配置文件。然后，您可以使用 Systems Manager 运行合规性扫描并查看合规

和不合规的实例。有关更多信息，请参阅 [将 Chef InSpec 配置文件与 Systems Manager Compliance 结合使用 \(p. 55\)](#)。

关于补丁合规性

在您使用 Patch Manager 在实例上安装补丁之后，会立即在控制台中或在一组 AWS CLI 命令或相应 Systems Manager API 操作的响应中提供合规性状态信息。

Note

如果您要向实例分配特定补丁合规性状态，可以使用 [put-compliance-items](#) CLI 命令或 [PutComplianceItems](#) API 操作。Amazon EC2 控制台不支持分配合规性状态。

对于每个补丁，将报告以下合规性状态值之一：

- **Installed**：补丁已安装，或者在实例上运行 AWS-RunPatchBaseline 文档时 Patch Manager 安装了补丁。
- **Installed_Other**：补丁不在基准中，但已安装在实例上。某人可能已手动安装补丁。
- **Missing**：基准中已批准补丁，但实例上未安装补丁。如果将 AWS-RunPatchBaseline 文档任务配置为扫描（而不是安装），系统将为扫描期间找到但未安装的补丁报告此状态。
- **Not_Applicable**：基准中已批准补丁，但实例上未安装使用补丁的服务或功能。例如，如果基准中已批准 Web 服务器服务的补丁，但实例上未安装该 Web 服务，则补丁将显示 **Not_Applicable**。
- **Failed**：基准中已批准补丁，但无法安装补丁。要解决此问题，请查看命令输出中是否有可帮助您理解此问题的信息。

查看补丁合规性报告

您可以使用控制台或 CLI 来查看补丁合规性报告数据。

Note

有关修复合规性问题的信息，请参阅 [修复合规性问题 \(p. 96\)](#)。

查看补丁合规性报告 (控制台)

查看补丁合规性报告

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择合规性。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择合规性。

3. 在对应的托管实例区域中，选择实例的名称以查看其详细的配置合规性报告。

查看补丁合规性报告 (CLI)

您可以通过在 CLI 中运行以下命令来查看补丁合规性详细信息。

[list-compliance-summaries](#)

根据您指定的筛选条件返回合规和不合规补丁状态的摘要计数。(API：[ListComplianceSummaries](#))

[list-resource-compliance-summaries](#)

返回资源级摘要计数。根据您指定的筛选条件，摘要包括有关合规和不合规状态的信息，以及详细的合规性项目严重性计数。(API：[ListResourceComplianceSummaries](#))

describe-patch-group-state

返回补丁组的高级聚合补丁合规性状态。(API : [DescribePatchGroupState](#))

describe-instance-patch-states-for-patch-group

返回指定补丁组中实例的高级补丁状态。(API : [DescribeInstancePatchStatesForPatchGroup](#))

有关使用 AWS CLI 配置修补以及如何查看补丁合规性详细信息的说明，请参阅 [Systems Manager Patch Manager 演练 \(p. 225\)](#)。

关于关联合规性

创建一个或多个 状态管理器 关联后，配置合规性将自动报告关联合规性状态。您不需要执行任何其他步骤即可查看这些状态。如果您想将特定关联合规性状态分配给某个实例，则可以使用 [PutComplianceItems](#) API 操作显式分配状态。您可以从 AWS CLI、AWS Tools for Windows PowerShell 或软件开发工具包使用此 API 操作。您目前无法使用 Amazon EC2 控制台分配合规性状态。

您可以在 Amazon EC2 控制台中的 Compliance Configuration 页面上查看关联合规性详细信息，您也可以使用以下 API 查看合规性详细信息：

Note

目前，配置合规性显示 Compliant 或 Non-compliant 的合规性状态和 Unspecified 的严重性。

- [ListComplianceSummaries](#)：根据您指定的筛选条件返回合规和不合规关联状态的摘要计数。
- [ListResourceComplianceSummaries](#)：返回资源级摘要计数。根据您指定的筛选条件，摘要包括有关合规和不合规状态的信息，以及 Unspecified 计数。

关于自定义合规性

您可以将合规性元数据分配给托管实例。然后此元数据可以与用于合规性报告目的的其他合规性数据聚合。例如，假设您的企业在您的托管实例上运行软件 X 版本 2.0、3.0 和 4.0。公司希望在版本 4.0 上实现标准化，这意味着运行版本 2.0 和 3.0 的实例将不合规。您可以使用 [PutComplianceItems](#) API 操作显式注释哪些托管实例在运行较旧版本的软件 X。目前您只能使用 AWS CLI、AWS Tools for Windows PowerShell 或软件开发工具包分配合规性元数据。以下 CLI 示例命令会将合规性元数据分配给一个托管实例并以要求的格式 Custom: 指定合规性类型。

```
aws ssm put-compliance-items --resource-id i-1234567890 --resource-  
type ManagedInstance --compliance-type Custom:SoftwareXCheck --  
execution-summary ExecutionTime=AnyStringToDenoteTimeOrDate, --items  
Id=Version2.0,Title=SoftwareXVersion,Severity=CRITICAL,Status=NON_COMPLIANT
```

然后合规经理可以查看摘要或创建有关哪些实例合规或不合规的报告。您最多可将 10 个不同的自定义合规性类型分配给一个实例。

有关如何创建自定义合规性类型并查看合规性数据的示例，请参阅[Systems Manager 配置合规性管理器演练 \(AWS CLI\) \(p. 97\)](#)。

修复合规性问题

您可以使用 Systems Manager Run Command 快速修复补丁和关联合规性问题。您可以将实例 ID 或 Amazon EC2 标签设为目标并执行 AWS-RefreshAssociation 文档或 AWS-RunPatchBaseline 文档。如果刷新关联或重新运行补丁基准未能解决合规性问题，则需要调查您的关联、补丁基准或实例配置，以了解 Run Command 不能解决问题的原因。有关运行命令的更多信息，请参阅[使用 Systems Manager Run Command 运行命令 \(p. 181\)](#)。

您也可以将 CloudWatch Events 配置为执行操作以响应配置合规性事件。例如，如果一个或多个实例未能安装重要补丁更新或运行安装反病毒软件的关联，则您可以将 CloudWatch 配置为在配置合规性事件发生时运行 AWS-RunPatchBaseline 文档或 AWS-RefreshAssociation 文档。使用以下过程将配置合规性配置为 CloudWatch 事件的目标。

将配置合规性配置为 CloudWatch 事件的目标

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. 在左侧导航窗格中，选择 Events，然后选择 Create rule。
3. 选择 Event Pattern。利用 Event Pattern，您可以构建将为 AWS Services 中的特定操作生成事件的规则。
4. 在 Service Name 字段中，选择 EC2 Simple Systems Manager (SSM)
5. 在 Event Type 字段中，选择 Configuration Compliance。
6. 选择 Add target。
7. 在 Select target type 列表中，选择 SSM Run Command。
8. 在 Document 列表中，选择在调用目标时要运行的 SSM 文档。例如，为不合规关联事件选择 AWS-RefreshAssociation，或为不合规补丁事件选择 AWS-RunPatchBaseLine。
9. 指定剩余字段和参数的信息。

Note

必需字段和参数的名称旁有一个星号 (*)。要创建目标，您必须为每个必需的参数或字段指定一个值。如果没有指定，系统将创建规则，但不执行规则。

10. 选择 Configure details 并完成向导。

Systems Manager 配置合规性管理器演练 (AWS CLI)

以下步骤为您演示了使用 `PutComplianceItems` API 操作将自定义合规性元数据分配给资源的过程。您也可以使用此 API 操作手动将补丁或关联合规性元数据分配给某个实例，如以下演练中所示。有关自定义合规性的更多信息，请参阅[关于自定义合规性 \(p. 96\)](#)。

将自定义合规性元数据分配给托管实例

1. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令，以便将自定义合规性元数据分配给某个实例。当前，仅支持 `ManagedInstance` 资源类型。

```
aws ssm put-compliance-items --resource-id Instance ID --resource-type
ManagedInstance --compliance-type Custom:User-defined string --execution-
summary ExecutionTime=User-defined time and/or date value --items Id=User-
defined ID,Title=User-defined title,Severity=One or more comma-separated
```



```
severities:CRITICAL,MAJOR,MINOR,INFORMATIONAL, or UNSPECIFIED,Status=COMPLIANT or  
NON_COMPLIANT
```

4. 重复上一步以将其他自定义合规性元数据分配给一个或多个实例。您也可以使用以下命令将补丁或关联合规性元数据手动分配给托管实例：

关联合规性元数据

```
aws ssm put-compliance-items --resource-id Instance ID --resource-type ManagedInstance  
--compliance-type Association --execution-summary ExecutionTime=User-defined time  
and/or date value --items Id=User-defined ID,Title=User-defined title,Severity=One  
or more comma-separated severities:CRITICAL,MAJOR,MINOR,INFORMATIONAL, or  
UNSPECIFIED,Status=COMPLIANT or NON_COMPLIANT,Details="{DocumentName=The SSM document  
for the association,DocumentVersion=A version number}"
```

补丁合规性元数据

```
aws ssm put-compliance-items --resource-id Instance ID --resource-type ManagedInstance  
--compliance-type Patch --execution-summary ExecutionTime=User-defined time and/  
or date value,ExecutionId=User-defined ID,ExecutionType=Command --items Id=for  
example, KB12345,Title=User-defined title,Severity=One or more comma-separated  
severities:CRITICAL,MAJOR,MINOR,INFORMATIONAL, or UNSPECIFIED,Status=COMPLIANT or  
NON_COMPLIANT,Details="{PatchGroup=Name of group,PatchSeverity=The patch severity, for  
example, CRITICAL}"
```

5. 执行以下命令，查看特定托管实例的合规性项目列表。使用筛选条件深入了解特定合规性数据。

```
aws ssm list-compliance-items --resource-ids Instance ID --resource-types  
ManagedInstance --filters One or more filters.
```

以下示例向您演示如何将此命令与筛选条件结合使用。

```
aws ssm list-compliance-items --resource-ids i-1234567890abcdef0 --resource-  
type ManagedInstance --filters Key=DocumentName,Values=AWS-RunPowerShellScript  
Key=Status,Values=NON_COMPLIANT,Type=NotEqual Key=Id,Values=cee20ae7-6388-488e-8be1-  
a88cc6c46dcc Key=Severity,Values=UNSPECIFIED
```

```
aws ssm list-resource-compliance-summaries --filters  
Key=OverallSeverity,Values=UNSPECIFIED
```

```
aws ssm list-resource-compliance-summaries --filters  
Key=OverallSeverity,Values=UNSPECIFIED Key=ComplianceType,Values=Association  
Key=InstanceId,Values=i-1234567890abcdef0
```

6. 执行以下命令查看合规性状态的摘要。使用筛选条件深入了解特定合规性数据。

```
aws ssm list-resource-compliance-summaries --filters One or more filters.
```

以下示例向您演示如何将此命令与筛选条件结合使用。

```
aws ssm list-resource-compliance-summaries --filters Key=ExecutionType,Values=Command
```

```
aws ssm list-resource-compliance-summaries --filters
  Key=AWS:InstanceInformation.PlatformType,Values=Windows
  Key=OverallSeverity,Values=CRITICAL
```

7. 执行以下命令查看某个合规性类型的合规资源和不合规资源的摘要计数。使用筛选条件深入了解特定合规性数据。

```
aws ssm list-compliance-summaries --filters One or more filters.
```

以下示例向您演示如何将此命令与筛选条件结合使用。

```
aws ssm list-compliance-summaries --filters
  Key=AWS:InstanceInformation.PlatformType,Values=Windows
  Key=PatchGroup,Values=TestGroup
```

```
aws ssm list-compliance-summaries --filters
  Key=AWS:InstanceInformation.PlatformType,Values=Windows
  Key=ExecutionId,Values=4adf0526-6aed-4694-97a5-145222f4c2b6
```


AWS Systems Manager 操作

AWS Systems Manager (以前称作 Amazon EC2 Systems Manager) 提供以下可对 AWS 资源执行操作的功能。

主题

- [AWS Systems Manager 自动化 \(p. 100\)](#)
- [AWS Systems Manager Run Command \(p. 171\)](#)
- [AWS Systems Manager Patch Manager \(p. 204\)](#)
- [AWS Systems Manager Maintenance Window \(p. 246\)](#)
- [AWS Systems Manager 状态管理器 \(p. 277\)](#)

AWS Systems Manager 自动化

Systems Manager Automation 是一项 AWS 托管的服务，能够简化常见实例和系统维护及部署任务。例如，您可在变更管理过程中使用 Automation，通过最新应用程序生成保持您的 Amazon Machine Image (AMI) 为最新状态。或者，假设您要在夜间创建数据库备份并将其上传到 Amazon S3 中。使用 Automation，您可以避免部署脚本以及直接对实例计划逻辑。您可以改为通过 Systems Manager Run Command 以及由 Automation 服务编排的 AWS Lambda 步骤运行维护活动。

使用 Automation 可以执行以下操作。

- 在您的 Amazon Machine Image (AMI) 中，通过可由您审核的简化且可重复的过程，预先安装和配置应用程序及代理。
- 构建工作流以配置和管理实例及 AWS 资源。
- 创建您的自定义工作流，或者使用由 AWS 维护的预定义工作流。
- 使用 Amazon CloudWatch Events 接收有关 Automation 任务和工作流的通知
- 使用 Amazon EC2 或 AWS Systems Manager 控制台监控 Automation 进度和执行详细信息。

内容

- [AWS Systems Manager 自动化概念 \(p. 100\)](#)
- [Automation 使用案例 \(p. 101\)](#)
- [Automation 快速入门 \(p. 103\)](#)
- [设置 Automation \(p. 107\)](#)
- [Systems Manager Automation 演练 \(p. 114\)](#)
- [使用 Automation 文档 \(p. 118\)](#)
- [Systems Manager Automation 示例 \(p. 133\)](#)
- [Automation 系统变量 \(p. 152\)](#)
- [排除 Systems Manager Automation 的故障 \(p. 160\)](#)

AWS Systems Manager 自动化概念

AWS Systems Manager 自动化使用以下概念。

概念	详细信息
自动化文档	Systems Manager 自动化文档定义自动化工作流程 (Systems Manager 在托管实例和 AWS 资源上执行的操作)。自动化包含几个预定义的自动化文档，您可以用它们来执行常见任务，例如重启一个或多个 Amazon EC2 实例，或创建 Amazon Machine Image (AMI)。文档使用 JavaScript Object Notation (JSON) 或 YAML，并包括您指定的步骤和参数。步骤按先后顺序运行。有关更多信息，请参阅 使用 Automation 文档 (p. 118) 。
自动化操作	自动化文档中定义的自动化工作流程中包含一个或多个步骤。每个步骤都与特定操作或插件相关。此操作确定本步的输入、行为和输出。步骤定义在 Automation 文档的 mainSteps 部分。有关更多信息，请参见 Systems Manager 自动化文档参考 (p. 334) 。
自动化队列	每个 AWS 账户可以同时运行 25 项自动化。如果您尝试运行的自动化多于此数量，Systems Manager 会将额外的执行添加到一个队列中，状态显示为“待处理”。自动化完成 (或达到最终状态) 后，队列中的第一个执行将启动。每个 AWS 账户可以在队列中加入 75 个自动化执行。

Automation 使用案例

本部分包括 AWS Systems Manager Automation 的常见使用案例。

执行常见 IT 任务

Automation 可以根据计划简化常见 IT 任务，如更改一个或多个实例的状态 (使用审批工作流程) 和管理实例状态。下面是一些示例：

- 使用 AWS-StopInstance 文档请求一个或多个 AWS Identity and Access Management (IAM) 用户批准实例停止操作。在获得批准后，Automation 将停止实例。
- 通过使用 Amazon CloudWatch Events 或使用 Maintenance Window 任务，使用 AWS-StopInstance 文档以按计划自动停止实例。例如，您可以配置 Automation 工作流程以在每个星期五晚上停止实例，然后在每个星期一早晨重新启动这些实例。
- 使用 AWS-UpdateCloudFormationStackWithApproval 文档来更新通过使用 CloudFormation 模板部署的资源。更新会应用新模板。在更新开始之前，您可以配置 Automation 以请求由一个或多个 IAM 用户批准。

安全地批量执行中断性任务

Systems Manager 包含可通过使用 EC2 标签帮助您确定大型目标实例组的功能以及可根据您定义的限制帮助实施更改的速度控制功能。

使用 AWS RestartInstanceWithApproval 文档以定位包含多个实例的 AWS 资源组。您可以配置 Automation 工作流程来使用速度控制功能。例如，您可以指定应同时重新启动的实例数量。您还可以指定在取消 Automation 工作流程之前允许的最大错误数。

简化复杂任务

Automation 为简化复杂任务提供了一键式自动化，例如创建黄金 Amazon Machine Image (AMI) 和恢复无法访问的 EC2 实例。下面是一些示例：

- 使用 AWS-UpdateLinuxAMI 和 AWS-UpdateWindowsAMI 文档通过源 AMI 来创建黄金 AMI。在应用更新前后，您可以运行自定义脚本。您还可以包含特定程序包或从安装中排除这些程序包。
- 使用 AWSSupport-ExecuteEC2Rescue 文档恢复受损的实例。出于各种原因，实例可能无法访问，包括网络配置错误、RDP 问题或防火墙设置。以前，对实例进行故障排除并重新获得其访问权限需要执行几十个手动步骤，然后才能重新获得访问权限。使用 AWSSupport-ExecuteEC2Rescue 文档，您可以通过指定实例 ID 并单击一个按钮来重新获得访问权限。

增强操作安全

使用委托管理，您可以限制或提升各种类型任务的用户权限。

借助委托管理，您可以提供对特定资源上特定任务的权限，而无需为用户授予访问资源的直接权限。这可以提升整体安全配置文件。例如，假定用户 1 没有重新启动 EC2 实例的权限，但您希望授权该用户执行此操作。那么不用为用户 1 授予直接权限，您可以执行以下操作：

- 创建一个 IAM 角色，并为其授予成功停止和启动 EC2 实例所需的权限。
- 创建 Automation 文档并在文档中嵌入该角色。(执行此操作最简单方法是自定义 AWS-RestartEC2Instance 文档并在文档中嵌入该角色，而不用分配 Automation 服务角色 [或代入角色])。
- 修改 User1 的 IAM 权限并授予用户运行本文档的权限。下面是如何修改文档的示例：

```
"Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:StartAutomationExecution",
      "Effect": "Allow",
      "Resource": [
        "ARN of the document you created in Step 2"
      ]
    }
  ]
}
```

User1 现在可以运行此文档并重新启动实例了。

分享最佳实践

Automation 允许您与组织中的其余人员分享最佳实践。

您可以在 Automation 文档中创建资源管理的最佳实践，并轻松地在 AWS 区域和组中共享这些文档。您也可以约束文档接受的参数值。下面给出了一个用于启动新 EC2 实例并限制用户仅指定特定 AMI ID 的 Automation 文档的示例：

Note

根据需要替换 AMI 值以及 MySecurityKeyName 和 IamInstanceProfileName 的值。

```
{
  "description": "Launch Allowed EC2 instances",
  "schemaVersion": "0.3",
  "assumeRole": "{ AutomationAssumeRole }",
  "parameters": {
    "AMIID": {
      "type": "String",
      "description": "Instance to value",
      "allowedValues": [ "ami-e3bb7399", "ami-55ef662f" ]
    },
    "AutomationAssumeRole": {
      "type": "String",
```

```
    "description": "(Optional) The ARN of the role that allows Automation to perform the  
actions on your behalf.",  
    "default": ""  
  },  
  },  
  "mainSteps": [  
    {  
      "name": "launchInstances",  
      "action": "aws:runInstances",  
      "timeoutSeconds": 1200,  
      "maxAttempts": 1,  
      "onFailure": "Abort",  
      "inputs": {  
        "ImageId": "{ { AMIID } }",  
        "InstanceType": "t2.micro",  
        "KeyName": "MySecurityKeyName",  
        "MinInstanceCount": 1,  
        "MaxInstanceCount": 1,  
        "IamInstanceProfileName": "ManagedInstanceProfile"  
      }  
    }  
  ]  
}
```

此文档可确保用户仅可通过指定的 AMI ID 来启动 t2.micro 实例。

Automation 快速入门

本部分包含可帮助您在几分钟内运行简单的 Systems Manager 自动化工作流程的演练。每项演练都提供一种不同方法来设置和执行 Automation 工作流程。我们建议您在 AWS Identity and Access Management (IAM) 中拥有管理员权限的测试环境中执行这些演练。

内容

- [快速入门第 1 步：以当前经过身份验证的用户身份运行 Automation 工作流程 \(p. 103\)](#)
- [快速入门第 2 步：通过使用 IAM 服务角色运行 Automation 工作流程 \(p. 104\)](#)
- [快速入门第 3 步：使用委托管理来增强自动化安全性 \(p. 104\)](#)

快速入门第 1 步：以当前经过身份验证的用户身份运行 Automation 工作流程

此演练向您展示如何通过使用 AWS-RestartEC2Instance 文档运行重新启动托管实例的自动化工作流程。工作流程在当前 IAM 用户的环境中运行。这意味着只要您有权运行 Automation 文档和此文档调用的任何操作，就无需配置其他 IAM 权限。如果您在 IAM 中拥有管理员权限，则有权运行此 Automation。

以当前经过身份验证的用户身份运行自动化文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择托管实例。
3. 复制要重新启动的另一个托管实例的实例 ID。
4. 在导航窗格中，选择自动化，然后选择执行自动化。
5. 在 Automation 文档列表中，选择 AWS-RestartEC2Instance。
6. 在文档详细信息部分，验证文档版本是否设置为 1 (默认)。
7. 对于执行模式和目标和速率控制部分，保留默认设置。
8. 在输入参数部分，在实例 ID 框粘贴一个或多个 ID。用逗号 (,) 分隔实例 ID。

Note

您可以复制粘贴实例 ID 的垂直列表 (用回车符分隔的 ID)，因为系统会自动分隔每个实例 ID。

9. 选择执行自动化。控制台将显示 Automation 执行的状态。

快速入门第 2 步：通过使用 IAM 服务角色运行 Automation 工作流程

此演练向您展示如何通过使用 AWS-RestartEC2Instance 文档运行重新启动托管实例的自动化工作流程。工作流程通过使用 IAM 服务角色 (或代入角色) 来运行自动化。服务角色代表您向 Automation 提供执行操作的服务权限。当您想要限制权限并以最小权限运行操作时，配置服务角色非常有用。例如，如果您想要限制用户对资源 (例如 EC2 实例) 的权限，但是要让用户运行用于执行一组允许的特定操作的自动化工作流程。在这种情况下，您可以创建具有更高权限的服务角色并允许用户运行自动化工作流程。

开始前的准备工作

在您完成以下过程之前，必须创建 IAM 服务角色并为 Automation 配置信任关系。有关更多信息，请参阅以下流程：[任务 1：为 Automation 创建服务角色 \(p. 110\)](#) 和 [任务 2：为 Automation 添加信任关系 \(p. 111\)](#)。

通过使用服务角色运行自动化文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择托管实例。
3. 复制要重新启动的另一个托管实例的实例 ID。
4. 在导航窗格中，选择自动化，然后选择执行自动化。
5. 在 Automation 文档列表中，选择 AWS-RestartEC2Instance。
6. 在文档详细信息部分，验证文档版本是否设置为 1 (默认)。
7. 对于执行模式和目标和速率控制部分，保留默认设置。
8. 在输入参数部分，在实例 ID 框粘贴一个或多个 ID。用逗号 (,) 分隔实例 ID。

Note

您可以复制粘贴实例 ID 的垂直列表 (用回车符分隔的 ID)，因为系统会自动分隔每个实例 ID。

9. 在 Automation 代入角色框中，粘贴 IAM 服务角色的 ARN。
10. 选择执行自动化。控制台将显示 Automation 执行的状态。

有关如何使用 Systems Manager Automation 的更多示例，请参阅 [Systems Manager Automation 演练 \(p. 114\)](#)。有关 Automation 入门的信息，请参阅[设置 Automation \(p. 107\)](#)。

快速入门第 3 步：使用委托管理来增强自动化安全性

当您运行 AWS Systems Manager 自动化时，默认情况下，自动化会在启动执行的 AWS Identity and Access Management (IAM) 用户的上下文中运行。也就是说，例如，如果您的 IAM 用户账户具有管理员权限，则自动化运行时也具有管理员权限，可完全访问针对自动化所配置的资源。作为最佳安全实践，我们建议您使用配置有 AmazonSSMAutomationRole 托管策略的 IAM 服务角色 (也称为代入角色) 运行自动化。使用 IAM 服务角色运行自动化称为委托管理。

如果使用服务角色，则允许针对 AWS 资源运行自动化，但运行自动化的用户对这些资源的访问受限 (或无法访问)。例如，您可以配置一个服务角色，并用它自动重启一个或多个 Amazon EC2 实例。自动化会重启实例，但服务角色不会授予用户访问这些实例的权限。

可以在您运行自动化的运行时中指定服务角色，也可以创建自定义自动化文档，直接在文档中指定服务角色。无论您是在运行时还是在自动化文档中指定了服务角色，服务都会在所指定服务角色的上下文中运行。如果您未指定服务角色，系统会在用户上下文中创建临时会话并运行自动化。

Note

对于预计运行时长超过 12 小时的自动化，必须指定服务角色。如果您在用户环境中启动了长时间运行的自动化，用户的临时会话会在 12 小时后过期。

委托管理可确保更高的安全性，还能更好地控制您的 AWS 资源。它还增强了审核体验，因为针对您的资源进行的操作是由集中的服务角色执行的，而不是由多个 IAM 账户执行。

为了恰当地说明委托管理在组织中的运作方式，本主题将带您演练以下任务，就好像这些任务是由组织中的三个人执行的：

- 创建测试 IAM 用户账户，名为 AutomationRestrictedOperator (管理员)
- 创建进行自动化所用的 IAM 服务角色 (管理员)
- 根据现有的自动化文档创建简单的自动化文档，用于指定服务角色 (SSM 文档作者)
- 以测试用户的身份运行自动化 (受限操作员)

在某些组织中，所有的三项任务都是由同一人员执行的。但在本例中，确定不同的角色可帮助您了解委托管理如何在复杂的组织中增强安全性。

Important

作为最佳安全实践，我们建议您始终使用服务角色来运行自动化，即使您是一名执行所有这些任务的管理员。

本节中的过程会链接到其他 AWS 指南中的主题或其他 Systems Manager 主题。我们建议您在 Web 浏览器的新标签页中打开指向其他主题的连接，这样就不会丢失本主题中的位置。

主题

- [创建测试用户账户 \(p. 105\)](#)
- [创建用于自动化的 IAM 服务角色 \(p. 106\)](#)
- [创建自定义自动化文档 \(p. 106\)](#)
- [运行自定义自动化文档 \(p. 107\)](#)

创建测试用户账户

本部分介绍如何创建具有受限权限的 IAM 测试用户账户。权限集允许用户运行自动化，但用户无法访问自动化的目标 AWS 资源。操作员也可以查看自动化的结果。首先，您需要创建自定义 IAM 权限策略，然后再创建用户账户并向它分配权限。

创建 IAM 测试用户

1. 创建名为 OperatorRestrictedPermissions 的权限策略。有关如何新建 IAM 权限策略的信息，请参阅 IAM User Guide 中的 [创建 IAM 策略 \(控制台\)](#)。在 JSON 选项卡上创建策略，并指定以下权限集。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeAutomationExecutions",
```



```
        "ssm:DescribeAutomationStepExecutions",
        "ssm:DescribeDocument",
        "ssm:GetAutomationExecution",
        "ssm:GetDocument",
        "ssm:ListDocuments",
        "ssm:ListDocumentVersions",
        "ssm:StartAutomationExecution"
    ],
    "Resource": "*"
  }
}
```

2. 新建名为 AutomationRestrictedOperator 的 IAM 用户账户。有关如何新建 IAM 用户的信息，请参阅 IAM User Guide 中的[创建 IAM 用户 \(控制台\)](#)。在提示时选择直接附加现有策略，然后选择刚创建的策略。
3. 记下用户名、密码和控制台登录链接。在本主题的后面部分，您将登录此账户。

创建用于自动化的 IAM 服务角色

以下过程会链接到其他主题，可帮助您创建服务角色，并配置自动化以信任此角色。

创建服务角色，并使自动化信任它

1. 创建自动化服务角色。有关信息，请参阅[任务 1：为 Automation 创建服务角色 \(p. 110\)](#)。
2. 记下服务角色的 Amazon 资源名称 (ARN)。您将在下一过程中指定此 ARN。
3. 配置信任策略，使自动化信任此服务角色。有关更多信息，请参阅[任务 2：为 Automation 添加信任关系 \(p. 111\)](#)。

创建自定义自动化文档

本部分介绍如何创建重启 Amazon EC2 实例的自定义自动化文档。AWS 提供用于重启实例的默认 SSM 文档，名为 AWS-RestartEC2Instance。以下过程会复制该文档的内容，演示如何在您自己创建的文档中输入服务角色。如果直接在文档中指定服务，则用户在执行文档时不需要 iam:PassRole 权限。如果没有 iam:PassRole 权限，用户无法在 AWS 中的其他位置使用此服务角色。

创建自定义自动化文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择创建文档。
4. 在名称字段中键入文档名称，如 Restart-EC2InstanceDemo。
5. 在文档类型列表中，选择自动化文档。
6. 在内容部分选择 JSON，然后粘贴以下内容。将 AssumeRoleARN 替换为您在上一过程中创建的服务角色的 ARN。

```
{
  "description": "Restart EC2 instances(s)",
  "schemaVersion": "0.3",
  "assumeRole": "service role ARN",
  "parameters": {
```

```
"InstanceId": {
  "type": "StringList",
  "description": "(Required) EC2 Instance to restart"
},
"mainSteps": [
  {
    "name": "stopInstances",
    "action": "aws:changeInstanceState",
    "inputs": {
      "InstanceIds": "{{ InstanceId }}",
      "DesiredState": "stopped"
    }
  },
  {
    "name": "startInstances",
    "action": "aws:changeInstanceState",
    "inputs": {
      "InstanceIds": "{{ InstanceId }}",
      "DesiredState": "running"
    }
  }
]
}
```

7. 选择创建文档。

运行自定义自动化文档

以下过程介绍了如何以受限操作员的身份（在本主题前面部分创建）运行您刚刚创建的文档。用户可以运行您之前创建的文档，是因为他们的 IAM 账户权限使他们能够看到并运行该文档。但用户无法登录此自动化将要重启的实例。

1. 在 <https://console.aws.amazon.com/ec2/> 中，复制您希望使用以下自动化重启的一个或多个实例的实例 ID。
2. 从 AWS 管理控制台中注销，然后使用测试用户账户重新登录您之前复制的控制台登录链接
3. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
4. 在导航窗格中，选择 Automation。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Automation。

5. 选择执行自动化。
6. 选择您在本主题前面创建的自定义自动化文档。
7. 在文档详细信息部分，验证文档版本是否设置为 1 (默认)。
8. 在执行模式部分中，选择一次执行整个自动化。
9. 保留目标和速率控制选项为禁用状态。
10. 在输入参数部分中，键入您要重启的一个或多个实例 ID，然后选择执行自动化。

执行详细信息介绍了自动化的状态。步骤 1 会将实例停止。步骤 2 会重启它们。

设置 Automation

安装 Automation 要求您验证用户对 Automation 服务的访问权限并配置角色，以便服务可在您的实例上执行操作。或者，您也可以配置 Automation 向 Amazon CloudWatch Events 发送事件。您还可以配置在发生特

定 CloudWatch 事件时要运行的 Automation 工作流程。这称为将 Automation 指定为 CloudWatch 事件的目标。

内容

- [配置对 Systems Manager Automation 的访问权限 \(p. 108\)](#)
- [为 Systems Manager Automation 配置 CloudWatch Events \(可选\) \(p. 113\)](#)
- [将 Automation 配置为 CloudWatch Events 目标 \(可选\) \(p. 113\)](#)

配置对 Systems Manager Automation 的访问权限

配置对 Systems Manager Automation 的访问权限需要完成以下任务。

1. 验证用户访问权限：验证您是否拥有运行 Automation 工作流程的权限。如果已为您的 AWS Identity and Access Management (IAM) 用户账户、组或角色分配管理员权限，则您可以访问 Systems Manager Automation。如果您没有管理员权限，则管理员必须通过向您的 IAM 账户、组或角色分配 AmazonSSMFullAccess 托管策略或提供类似权限的策略来向您授予权限。
2. 通过创建和分配实例配置文件角色配置实例访问权限 (必需)：运行 Automation 工作流程的每个实例都需要一个 IAM 实例配置文件角色。此角色为 Automation 提供在您的实例上执行操作的权限，例如执行命令或启动和停止服务。如果之前已经为 Systems Manager 创建了实例配置文件角色 (如配置对 Systems Manager 的访问权限主题中的[任务 2：为 Systems Manager 创建实例配置文件 \(p. 10\)](#)所述)，则您可以为 Automation 使用同一个实例配置文件角色。有关如何将此角色附加到现有实例的信息，请参阅 Amazon EC2 用户指南中的[将 IAM 角色附加到实例](#)。

Note

Automation 之前需要您指定服务角色 (或代入 角色)，以便服务有权代表您执行操作。Automation 不再需要此角色，因为服务现在通过使用已调用执行的用户的上下文来操作。不过，以下情况仍需要您为 Automation 指定服务角色：

- 当您想限制用户对资源的权限，但希望用户运行需要更高权限的自动化工作流程时。在此方案中，您可以创建具有更高权限的服务角色并允许用户运行此工作流程。
- 运行时长预计将超过 12 小时的操作需要一个服务角色。

如果您需要为 Automation 创建一个服务角色和一个实例配置文件角色，您可以使用以下方法之一。

主题

- [方法 1：使用 AWS CloudFormation 为 Automation 配置角色 \(p. 108\)](#)
- [方法 2：使用 IAM 为 Automation 配置角色 \(p. 110\)](#)

方法 1：使用 AWS CloudFormation 为 Automation 配置角色

您可以通过 AWS CloudFormation 模板为 Automation 创建一个 IAM 实例配置文件角色和一个服务角色。

在您创建实例配置文件角色之后，必须将其分配到计划使用 Automation 配置的实例。有关如何将角色分配到现有实例的信息，请参阅 Amazon EC2 用户指南 中的[将 IAM 角色连接到实例](#)。有关在创建新实例时如何分配角色的信息，请参阅配置对 Systems Manager 的访问权限主题中的[任务 3：创建使用 Systems Manager 实例配置文件的 Amazon EC2 实例 \(p. 11\)](#)。

Note

您还可以在 Automation 文档中使用这些角色及其 Amazon 资源名称 (ARN)，例如 AWS-UpdateLinuxAmi 文档。在 Automation 文档中使用这些角色或其 ARN，可以让 Automation 在您的

托管实例上执行操作、启动新实例以及代表您执行操作。要查看示例，请参阅 [Automation CLI 演练：修补 Linux AMI \(p. 116\)](#)。

主题

- [使用 AWS CloudFormation 创建实例配置文件角色和服务角色 \(p. 109\)](#)
- [复制 Automation 的角色信息 \(p. 109\)](#)

使用 AWS CloudFormation 创建实例配置文件角色和服务角色

使用 AWS CloudFormation，通过以下过程为 Systems Manager Automation 创建所需的 IAM 角色。

创建所需的 IAM 角色

1. 选择 Launch Stack 按钮。该按钮会打开 AWS CloudFormation 控制台并用 Systems Manager Automation 模板的 URL 填充 Specify an Amazon S3 template URL 字段。

Note

选择 View 以查看模板。

查看	启动
查看	

2. 选择 Next。
3. 在 Specify Details 页面上的 Stack Name 字段中，选择保留默认值或指定您自己的值。选择 Next。
4. 在 Options 页面上，您无需进行任何选择。选择 Next。
5. 在 Review 页面上，向下滚动并选择 I acknowledge that AWS CloudFormation might create IAM resources 选项。
6. 选择 Create。

大约三分分钟左右，AWS CloudFormation 会显示 CREATE_IN_PROGRESS 状态。创建堆栈并且您的角色准备好使用之后，状态会变为 CREATE_COMPLETE。

复制 Automation 的角色信息

使用以下过程从 AWS CloudFormation 控制台复制关于实例配置文件角色和 Automation 服务角色的信息。当您运行 Automation 文档时，必须指定这些角色。

Note

如果您运行 AWS-UpdateLinuxAmi 或 AWS-UpdateWindowsAmi 文档，则无需使用此过程来复制角色信息。这些文档已将必需角色指定为默认值。这些文档中指定的角色使用 IAM 托管策略。

复制角色名称

1. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
2. 选中您在上一个过程中创建的 Automation 堆栈旁的复选框。
3. 选择 Resources 选项卡。
4. Resources 表在 Logical ID 列中包括三个项：AutomationServiceRole、ManagedInstanceProfile 和 ManagedInstanceRole。
5. 复制 ManagedInstanceProfile 的 Physical ID。物理 ID 类似于 **Automation-ManagedInstanceProfile-1a2b3c4**。这是您的实例配置文件角色的名称。
6. 在文本文件中粘贴实例配置文件角色以供以后使用。

7. 为 AutomationServiceRole 选择 Physical ID 链接。IAM 控制台将打开并显示 Automation 服务角色的摘要。
8. 复制 Role ARN 旁边的 Amazon 资源名称 (ARN)。ARN 类似于：arn:aws:iam::12345678:role/Automation-AutomationServiceRole-1A2B3C4D5E
9. 将 ARN 粘贴到文本文件供以后使用。

您已完成了为 Automation 配置所需的角色。现在，您可在 Automation 文档中使用实例配置文件角色和 Automation 服务角色 ARN。有关更多信息，请参阅 [Automation 控制台演练：修补 Linux AMI \(p. 114\)](#) 和 [Automation CLI 演练：修补 Linux AMI \(p. 116\)](#)。

方法 2：使用 IAM 为 Automation 配置角色

配置对 Systems Manager Automation 的访问权限需要完成以下任务。

1. 验证用户访问权限：验证您是否拥有运行 Automation 工作流程的权限。如果已为您的 AWS Identity and Access Management (IAM) 用户账户、组或角色分配管理员权限，则您可以访问 Systems Manager Automation。如果您没有管理员权限，则管理员必须通过向您的 IAM 账户、组或角色分配 AmazonSSMFullAccess 托管策略或提供类似权限的策略来向您授予权限。
2. 通过创建和分配实例配置文件角色配置实例访问权限 (必需)：运行 Automation 工作流程的每个实例都需要一个 IAM 实例配置文件角色。此角色为 Automation 提供在您的实例上执行操作的权限，例如执行命令或启动和停止服务。如果之前已经为 Systems Manager 创建了实例配置文件角色 (如配置对 Systems Manager 的访问权限主题中的 [任务 2：为 Systems Manager 创建实例配置文件 \(p. 10\)](#) 所述)，则您可以为 Automation 使用同一个实例配置文件角色。有关如何将此角色附加到现有实例的信息，请参阅 Amazon EC2 用户指南中的 [将 IAM 角色附加到实例](#)。

Note

Automation 之前需要您指定服务角色 (或代入角色)，以便服务有权代表您执行操作。Automation 不再需要此角色，因为服务现在通过使用已调用执行的用户的上下文来操作。不过，以下情况仍需要您为 Automation 指定服务角色：

- 当您想限制用户对资源的权限，但希望用户运行需要更高权限的自动化工作流程时。在此方案中，您可以创建具有更高权限的服务角色并允许用户运行此工作流程。
- 运行时长预计将超过 12 小时的操作需要一个服务角色。

如果您需要为 Systems Manager Automation 创建一个实例配置文件角色和一个服务角色，请完成以下任务。

任务

- [任务 1：为 Automation 创建服务角色 \(p. 110\)](#)
- [任务 2：为 Automation 添加信任关系 \(p. 111\)](#)
- [任务 3：将 iam:PassRole 策略附加到您的 Automation 角色 \(p. 112\)](#)
- [任务 4：配置用户对 Automation 的访问权限 \(p. 112\)](#)
- [任务 5：创建实例配置文件角色 \(p. 112\)](#)

任务 1：为 Automation 创建服务角色

使用以下步骤为 Systems Manager Automation 创建一个服务角色 (或代入角色)。

Note

您还可以在 Automation 文档中使用这些角色及其 Amazon 资源名称 (ARN)，例如 AWS-UpdateLinuxAmi 文档。在 Automation 文档中使用这些角色或其 ARN，可以让 Automation 在您的

托管实例上执行操作、启动新实例以及代表您执行操作。要查看示例，请参阅 [Automation CLI 演练：修补 Linux AMI \(p. 116\)](#)。

创建 IAM 角色并允许 Automation 代入该角色

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择 Roles，然后选择 Create Role。
3. 在 Select type of trusted entity 页面上的 AWS Service 下，选择 EC2。
4. 在选择您的使用案例部分，选择 EC2，然后选择下一步：权限。
5. 在附加的权限策略页面中，搜索 AmazonSSMAutomationRole 策略，选择它，然后选择下一步：审核。
6. 在 Review 页面上，在 Role name 框中键入名称，然后键入描述。
7. 选择 Create role。系统将让您返回到 Roles 页。
8. 在角色页面中，选择刚刚创建的角色以打开摘要页面。记下 Role Name 和 Role ARN。在后面的步骤中，当您将 iam:PassRole 策略附加到您的 IAM 账户时，将指定该角色 ARN。您还可以在 Automation 文档中指定角色名称和 ARN。

使摘要页面保持打开状态。

Note

AmazonSSMAutomationRole 策略会将 Automation 角色权限分配给您账户内 AWS Lambda 函数的子集。这些函数以“Automation”开头。如果您计划通过 Lambda 函数使用 Automation，则 Lambda ARN 必须使用以下格式：

```
"arn:aws:lambda:*:*:function:Automation"
```

如果现有 Lambda 函数的 ARN 未使用此格式，您还必须为 Automation 角色附加额外的 Lambda 策略，例如 AWSLambdaRole 策略。其他策略或角色必须提供对 AWS 账户内 Lambda 函数更宽泛的访问权限。

任务 2：为 Automation 添加信任关系

使用以下过程配置信任 Automation 的服务角色策略。

为 Automation 添加信任关系

1. 在刚刚创建的角色摘要页面上，选择信任关系选项卡，然后选择编辑信任关系。
2. 按照以下示例中所添加 "ssm.amazonaws.com"：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "ssm.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. 选择 Update Trust Policy。
4. 使摘要页面保持打开状态。

任务 3：将 iam:PassRole 策略附加到您的 Automation 角色

使用以下过程将 iam:PassRole 策略附加到您的 Automation 服务角色。这使 Automation 服务能够在运行 Automation 工作流程时将角色传递给其他服务或 Systems Manager 功能。

将 iam:PassRole 策略附加到您的 Automation 角色

1. 在刚刚创建的角色摘要页面中，选择权限选项卡。
2. 选择添加内联策略。
3. 在创建策略页面上，选择可视化编辑器选项卡。
4. 选择服务，然后选择 IAM。
5. 选择选择操作。
6. 在筛选操作文本框中，键入 PassRole，然后选择 PassRole 选项。
7. 选择资源。确保已选择 Specific，然后选择添加 ARN。
8. 在 Specify ARN for role 字段中，粘贴您在任务 1 结束时复制的自动化角色 ARN。系统会自动填充 Account 和 Role name with path 字段。
9. 选择 Add。
10. 选择查看策略。
11. 在查看策略页面上键入一个名称，然后选择创建策略。

任务 4：配置用户对 Automation 的访问权限

如果已为您的 AWS Identity and Access Management (IAM) 用户账户、组或角色分配管理员权限，则您可以访问 Systems Manager Automation。如果您没有管理员权限，则管理员必须通过向您的 IAM 账户、组或角色分配 AmazonSSMFullAccess 托管策略或提供类似权限的策略来向您授予权限。

使用以下过程配置用户账户，以使用 Automation。您选择的用户账户会拥有配置并运行自动化的权限。如果您需要创建新的用户账户，请参阅 IAM User Guide 中的[在您的 AWS 账户中创建 IAM 用户](#)。

配置用户访问权限并将 iam:PassRole 策略附加到用户账户

1. 在 IAM 导航窗格中，选择用户，然后选择您要配置的用户账户。
2. 在权限选项卡上的策略列表中，验证其中是否列出了 **AmazonSSMFullAccess** 策略或者授予账户访问 Systems Manager 权限的类似策略。
3. 选择添加内联策略。
4. 在 Set Permissions 页面上，选择 Policy Generator，然后选择 Select。
5. 验证 Effect 是否已设置为 Allow。
6. 从 AWS Services 中选择 AWS Identity and Access Management。
7. 从操作中，选择 PassRole。
8. 在 Amazon 资源名称 (ARN) 字段中，粘贴您在任务 1 结束时复制的 Automation 服务角色的 ARN。
9. 选择 Add Statement，然后选择 Next Step。
10. 在 Review Policy 页面上，选择 Apply Policy。

任务 5：创建实例配置文件角色

运行 Automation 工作流程的每项实例都需要一个 IAM 实例配置文件角色。此角色为 Automation 提供在您的实例上执行操作的权限，例如执行命令或启动和停止服务。如果之前已经为 Systems Manager 创建了实例

配置文件角色 (如配置对 Systems Manager 的访问权限主题中的[任务 2：为 Systems Manager 创建实例配置文件 \(p. 10\)](#)所述)，则您可以为 Automation 使用同一个实例配置文件角色。如果您尚未按该主题中所述的方法创建实例配置文件角色，请立即创建一个。有关如何将此角色附加到现有实例的信息，请参阅 Amazon EC2 用户指南中的[将 IAM 角色附加到实例](#)。

您已完成了为 Automation 配置所需的角色。现在，您可在 Automation 文档中使用实例配置文件角色和 Automation 服务角色 ARN。有关更多信息，请参阅 [Automation 控制台演练：修补 Linux AMI \(p. 114\)](#) 和 [Automation CLI 演练：修补 Linux AMI \(p. 116\)](#)。

为 Systems Manager Automation 配置 CloudWatch Events (可选)

您可以配置 Amazon CloudWatch Events 来向您通知 Systems Manager Automation 事件。例如，您可以配置 CloudWatch Events 在 Automation 步骤成功或失败时发送通知。您还可以配置 CloudWatch Events 在 Automation 工作流成功或失败时发送通知。使用以下过程配置 CloudWatch Events 发送有关 Automation 事件的通知。

为 Automation 配置 CloudWatch Events

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. 在左侧导航窗格中选择 Events，然后选择 Create rule。
3. 在 Event Source 下，验证已选中 Event Pattern。
4. 在 Service Name 字段中，选择 EC2 Simple Systems Manager (SSM)
5. 在 Event Type 字段中，选择 Automation。
6. 选择您要接收通知的详细类型和状态，然后选择 Add targets。
7. 在 Select target type 列表中，选择目标类型。有关不同目标类型的信息，请参阅对应的 AWS 帮助文档。
8. 选择 Configure details。
9. 指定规则详细信息，然后选择 Create rule。

下次运行 Automation 时，CloudWatch Events 会将事件详细信息发送到您指定的目标。

将 Automation 配置为 CloudWatch Events 目标 (可选)

您可通过指定 Automation 文档作为 Amazon CloudWatch 事件的目标来启动 Automation 工作流程。您可以按计划或者在特定 AWS 系统事件发生时启动工作流程。例如，假设您创建一个名为 BootStrapInstances 的 Automation 文档，该文档在实例启动时在实例上安装软件。要指定 BootStrapInstances 文档 (和对应的工作流程) 作为 CloudWatch 事件目标，请先创建新的 CloudWatch Events 规则。(以下是示例规则：Service name：EC2，Event Type：EC2 实例状态更改通知，Specific state(s)：正在运行，Any instance。)然后，您可使用以下过程指定 BootStrapInstances 文档作为事件目标。当新实例启动时，系统将运行工作流程并安装软件。

有关创建 Automation 文档的信息，请参阅[使用 Automation 文档 \(p. 118\)](#)。

使用以下过程配置 Automation 工作流程作为 CloudWatch 事件目标。

配置 Automation 作为 CloudWatch 事件目标

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. 在左侧导航窗格中，选择 Events，然后选择 Create rule。
3. 选择 Event Pattern 或 Schedule。利用 Event Pattern，您可以构建将为 AWS Services 中的特定操作生成事件的规则。利用 Schedule，您可以构建将根据您指定的计划使用 cron 格式生成事件的规则。

4. 选择要创建的规则的其余选项，然后选择 Add target。
5. 在 Select target type 列表中，选择 SSM Automation。
6. 在 Document 列表中，选择在调用目标时要运行的 SSM 文档。
7. 展开 Configure document version，然后选择一个版本。\$DEFAULT 已显式设置为 Systems Manager 中的默认文档。您可以选择特定版本，也可以使用最新版本。
8. 展开 Configure automation parameter(s)，保留默认参数值 (如果可用) 或输入您自己的值。

Note

所需参数的参数名旁有一个星号 (*)。要创建目标，您必须为每个所需的参数指定一个值。如果没有指定，系统将创建规则，但不运行规则。

9. 在权限部分中，选择一个选项。CloudWatch 使用此角色启动 Automation 工作流程。
10. 选择 Configure details 并完成向导。

Systems Manager Automation 演练

以下演练可帮助您使用预定义的 Automation 文档开始使用 Systems Manager Automation。

您必须先配置 Automation 角色和权限，然后才能开始使用。有关更多信息，请参阅 [设置 Automation \(p. 107\)](#)。有关创建自定义 Automation 文档的信息，请参阅 [演练：创建一个 Automation 文档 \(p. 128\)](#)。

Warning

如果您从正在运行的实例创建 AMI，则存在风险，实例的凭证、敏感数据或其他机密信息可能会记录到新映像中。创建 AMI 时请务必小心。

演练

- [Automation 控制台演练：修补 Linux AMI \(p. 114\)](#)
- [Automation CLI 演练：修补 Linux AMI \(p. 116\)](#)

Automation 控制台演练：修补 Linux AMI

此 Systems Manager Automation 演练将介绍如何使用控制台和 Systems Manager AWS-UpdateLinuxAmi 文档来通过您指定的最新版本的程序包自动修补 Linux AMI。您可以使用此演练更新以下任一 Linux 版本：Ubuntu、CentOS、RHEL、SLES 或 Amazon Linux AMI。AWS-UpdateLinuxAmi 文档也能自动安装其他具体站点相关的程序包和配置。

演练显示如何在运行时为 AWS-UpdateLinuxAmi 文档指定参数。如果您希望向自动化添加步骤或指定默认值，您可以使用 AWS-UpdateLinuxAmi 文档作为模板。

有关使用 Systems Manager 文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。有关您可以添加到文档的操作的信息，请参阅 [Systems Manager 自动化文档参考 \(p. 334\)](#)。

在运行 AWS-UpdateLinuxAmi 文档时，Automation 会执行以下任务。

1. 从 Linux AMI 启动临时 Amazon EC2 的实例。使用安装 SSM 代理的用户数据脚本配置实例。SSM 代理运行从 Systems Manager Run Command 远程发送的脚本。
2. 通过执行以下操作更新实例：
 - a. (可选) 在实例上调用户提供的更新前脚本。
 - b. 更新实例上的 AWS 工具 (如果有)。
 - c. 使用本地程序包管理器，更新实例上的分配程序包。

- d. (可选) 在实例上调用用户提供的更新后脚本。
3. 停止临时实例。
4. 从已停止的实例创建新 AMI。
5. 终止实例。

在 Automation 成功完成此工作流程后，即可在 AMI 页面上的控制台中使用新 AMI。

Important

如果使用 Automation 从一个实例创建 AMI，请注意，实例上的证书、密码、数据或者其他保密信息都会记录在新映像中。在从实例创建 AMI 时，请务必小心。

在您开始使用 Automation 时，请注意以下限制。

- Automation 不执行资源清除。在示例工作流中，当您的工作流步骤达到最终的实例终止步骤之前停止时，您可能需要手动停止实例或者禁用 Automation 工作流运行期间启动的服务。
- 如果您将用户数据用于 Automation，则用户数据必须为 base-64 编码。
- Automation 保留执行记录 30 天。
- Systems Manager 和 Automation 具有以下[服务限制](#)。

开始前的准备工作

创建 AWS Identity and Access Management (IAM) 实例配置文件角色和 Automation 服务角色 (或代入角色)。有关这些角色以及如何从 AWS CloudFormation 模板快速创建角色的更多信息，请参阅[方法 1：使用 AWS CloudFormation 为 Automation 配置角色 \(p. 108\)](#)。

我们建议您在开始前同时收集以下消息。

- 要更新的 AMI 的源 ID。
- (可选) 在更新前应用要运行的脚本的 URL。
- (可选) 在更新后应用要运行的脚本的 URL。
- (可选) 要更新的特定程序包的名称。默认情况下，Automation 会更新所有程序包。
- (可选) 要从更新中排除的特定程序包名称。

Note

默认情况下，当 Automation 运行 AWS-UpdateLinuxAmi 文档时，系统会在默认 VPC (172.30.0.0/16) 中创建一个临时实例。如果您删除了默认 VPC，会收到以下错误：
`VPC not defined 400`
要解决此问题，您必须复制 AWS-UpdateLinuxAmi 文档并指定子网 ID。有关更多信息，请参阅[VPC not defined 400 \(p. 161\)](#)。

使用 Automation 创建经过修补的 AMI ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Automation。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Automation。

3. 选择执行自动化。

4. 在自动化文档列表中，选择 AWS-UpdateLinuxAmi。
5. 在文档详细信息部分，确认文档版本是否设置为 1。
6. 在执行模式部分中，选择一次执行整个自动化。
7. 保留目标和速率控制选项为禁用状态。
8. 在 Input parameters 部分，输入在准备工作部分收集的信息。
9. 选择执行自动化。控制台将显示 Automation 执行的状态。

在工作流完成后，请从更新后的 AMI 启动测试实例以验证更改。

Note

如果工作流有任何步骤失败，则 Automation Executions 页面上会列出有关失败的信息。工作流设计为在成功完成所有任务后终止临时实例。如果步骤失败，系统可能不会终止实例。因此，如果某个步骤失败，请手动终止临时实例。

Automation CLI 演练：修补 Linux AMI

本 Systems Manager Automation 演练将介绍如何使用 AWS CLI 和 Systems Manager AWS-UpdateLinuxAmi 文档自动修补 Linux AMI。您可以使用此演练更新以下任一 Linux 版本：Ubuntu、CentOS、RHEL、SLES 或 Amazon Linux AMI。AWS-UpdateLinuxAmi 文档也能自动安装其他具体站点相关的程序包和配置。

在运行 AWS-UpdateLinuxAmi 文档时，Automation 会执行以下任务。

1. 从 Linux AMI 启动临时 Amazon EC2 的实例。使用安装 SSM 代理的用户数据脚本配置实例。SSM 代理运行从 Systems Manager Run Command 远程发送的脚本。
2. 通过执行以下操作更新实例：
 - a. 在实例上调用用户提供的更新前脚本。
 - b. 更新实例上的 AWS 工具 (如果有)。
 - c. 使用本地程序包管理器，更新实例上的分配程序包。
 - d. 在实例上调用用户提供的更新后脚本。
3. 停止临时实例。
4. 从已停止的实例创建新 AMI。
5. 终止实例。

在 Automation 成功完成此工作流程后，即可在 AMI 页面上的控制台使用新 AMI。

Important

如果使用 Automation 从一个实例创建 AMI，请注意，实例上的证书、密码、数据或者其他保密信息都会记录在新映像中。在从实例创建 AMI 时，请务必小心。

在您开始使用 Automation 时，请注意以下限制。

- Automation 不执行资源清除。在示例工作流中，当您的工作流步骤达到最终的实例终止步骤之前停止时，您可能需要手动停止实例或者禁用 Automation 工作流运行期间启动的服务。
- 如果您将用户数据用于 Automation，则用户数据必须为 base-64 编码。
- Automation 保留执行记录 30 天。
- Systems Manager 和 Automation 具有以下[服务限制](#)。

开始前的准备工作

创建 AWS Identity and Access Management (IAM) 实例配置文件角色和 Automation 服务角色 (或代入角色)。有关这些角色以及如何从 AWS CloudFormation 模板快速创建角色的更多信息，请参阅 [方法 1：使用 AWS CloudFormation 为 Automation 配置角色 \(p. 108\)](#)。

我们建议您同时收集 AMI 的源 ID 以进行更新。

Note

默认情况下，当 Automation 运行 AWS-UpdateLinuxAmi 文档时，系统会在默认 VPC (172.30.0.0/16) 中创建一个临时实例。如果您删除了默认 VPC，会收到以下错误：
VPC not defined 400
要解决此问题，您必须复制 AWS-UpdateLinuxAmi 文档并指定子网 ID。有关更多信息，请参阅 [VPC not defined 400 \(p. 161\)](#)。

使用 Automation 创建经过修补的 AMI

1. 将 AWS CLI [下载](#)到本地计算机上。
2. 运行以下命令以运行 AWS-UpdateLinuxAmi 文档和自动化工作流程。在参数部分，请指定您的 Automation 角色、AMI 源 ID 和 Amazon EC2 实例配置文件角色。

```
aws ssm start-automation-execution \  
--document-name "AWS-UpdateLinuxAmi" \  
--parameters \  
SourceAmiId=ami-e6d5d2f1
```

该命令将会返回执行 ID。请将该 ID 复制到剪贴板。您将使用该 ID 查看工作流的状态。

```
{  
  "AutomationExecutionId": "ID"  
}
```

3. 要使用 CLI 查看工作流程的执行情况，请运行以下命令：

```
aws ssm describe-automation-executions
```

4. 要查看有关执行进度的详细信息，请运行以下命令。

```
aws ssm get-automation-execution --automation-execution-id ID
```

更新过程可能需要 30 分钟或者更久。

Note

您也可以在控制台中监控工作流程的状态。在执行列表中，选择您刚才运行的执行，然后选择步骤选项卡。该选项卡将显示 workflow 操作的状态。

在工作流完成后，请从更新后的 AMI 启动测试实例以验证更改。

Note

如果工作流有任何步骤失败，则 Automation Executions 页面上会列出有关失败的信息。工作流设计为在成功完成所有任务后终止临时实例。如果步骤失败，系统可能不会终止实例。因此，如果某个步骤失败，请手动终止临时实例。

其他 Automation CLI 示例

您可使用以下任务管理 Automation 执行的其他方面。

停止执行

运行以下命令可停止工作流程。该命令不会终止相关实例。

```
aws ssm stop-automation-execution --automation-execution-id ID
```

创建 Automation 文档的版本

您无法更改现有自动化文档，但可以使用以下命令创建一个新版本：

```
aws ssm update-document --name "patchWindowsAmi" --content file:///Users/test-user/  
Documents/patchWindowsAmi.json --document-version "\$LATEST"
```

运行以下命令可查看关于现有文档版本的详细信息：

```
aws ssm list-document-versions --name "patchWindowsAmi"
```

此命令会返回如下信息：

```
{  
  "DocumentVersions": [  
    {  
      "IsDefaultVersion": false,  
      "Name": "patchWindowsAmi",  
      "DocumentVersion": "2",  
      "CreateDate": 1475799950.484  
    },  
    {  
      "IsDefaultVersion": false,  
      "Name": "patchWindowsAmi",  
      "DocumentVersion": "1",  
      "CreateDate": 1475799931.064  
    }  
  ]  
}
```

运行以下命令可更新执行的默认版本。只有在您明确地将默认执行版本设置为新版本时，该版本才会出现变化。创建新的文档版本不会更改默认版本。

```
aws ssm update-document-default-version --name patchWindowsAmi --document-version 2
```

删除文档

运行以下命令可删除自动化文档：

```
aws ssm delete-document --name patchWindowsAMI
```

使用 Automation 文档

Systems Manager Automation 文档定义 Systems Manager 在托管实例和 AWS 资源上执行的操作。文档使用 JavaScript Object Notation (JSON) 或 YAML，并包括您指定的步骤和参数。步骤按先后顺序运行。

Automation 文档是类型为 Automation 的 Systems Manager 文档，与 Command 和 Policy 文档相对。Automation 文档当前支持 0.3 版架构。Command 和 Policy 文档使用 1.2 或 2.0 版架构。

Note

要查看有关可以在 Systems Manager Automation 文档中指定的操作或插件的信息，请参阅 [Systems Manager 自动化文档参考 \(p. 334\)](#)。要查看有关所有其他 SSM 文档插件的信息，请参阅 [SSM 文档插件参考 \(p. 312\)](#)。

内容

- [使用预定义的 Automation 文档 \(p. 119\)](#)
- [创建动态自动化工作流程 \(p. 125\)](#)
- [演练：创建一个 Automation 文档 \(p. 128\)](#)

使用预定义的 Automation 文档

为了帮助您快速入门，Systems Manager 提供以下预定义 Automation 文档。这些文档由 Amazon Web Services 维护。

Note

任何名称中包含 WithApproval 的文档均表示该文档包含 [aws:approve \(p. 338\)](#) 操作。此操作会临时暂停自动化执行，直至指定委托人批准或拒绝操作。在达到所需批准数后，Automation 执行将恢复。

您可在 Systems Manager 控制台中查看这些文档的 JSON。

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择一个文档，然后选择查看详细信息。
4. 选择内容选项卡。

文档名称	目的
AWS-DeleteCloudFormationStack AWS-DeleteCloudFormationStackWithApproval	使用此文档删除 AWS CloudFormation 堆栈。您必须指定堆栈 ID。
AWS-RestartEC2Instance AWS-RestartEC2InstanceWithApproval	使用此文档重启一个或多个 Amazon EC2 实例 (Windows 或 Linux)。使用引号和逗号分隔实例 ID。例如，["i-0123abcd0123abcd0", "i-abcd0123abcd0123a"]。如果希望从自定义自动化文档中更改一个或多个实例的状态，请在文档中指定 aws:changeInstanceState (p. 340) 操作。
AWS-StartEC2Instance AWS-StartEC2InstanceWithApproval	使用此文档启动一个或多个 Amazon EC2 实例 (Windows 或 Linux)。使用引号和逗号分隔实例 ID。例如，["i-0123abcd0123abcd0", "i-abcd0123abcd0123a"]。如果希望从自定义自动化文档中更改一个或多个实例的状态，请在文档中指定 aws:changeInstanceState (p. 340) 操作。
AWS-StopEC2Instance	使用此文档停止一个或多个 Amazon EC2 实例 (Windows 或 Linux)。使用引号和逗号分隔

文档名称	目的
AWS-StopEC2InstanceWithApproval	实例 ID。例如，["i-0123abcd0123abcd0", "i-0123abcd0123abcd0123a"]。如果希望从自定义自动化文档中更改一个或多个实例的状态，请在文档中指定 aws:changeInstanceState (p. 340) 操作。
AWS-TerminateEC2Instance AWS-TerminateEC2InstanceWithApproval	使用此文档终止一个或多个 Amazon EC2 实例 (Windows 或 Linux)。使用引号和逗号分隔实例 ID。例如，["i-0123abcd0123abcd0", "i-0123abcd0123abcd0123a"]。如果希望从自定义自动化文档中更改一个或多个实例的状态，请在文档中指定 aws:changeInstanceState (p. 340) 操作。
AWS-UpdateCloudFormationStack AWS-UpdateCloudFormationStackWithApproval	通过此文档使用指定的模板对现有堆栈执行更新操作。您必须指定指向模板的 URL，该模板用于更新堆栈。这些文档运行 AWS Lambda 函数。您必须提供一个 IAM 角色的 Amazon 资源名称 (ARN)，供 Lambda 运行函数使用。
AWS-UpdateLinuxAmi	使用此文档自动执行 image-maintenance 任务。有关更多信息，请参阅 演练：使用 AWS-UpdateLinuxAmi 自定义和更新 Linux AMI (p. 120)。
AWS-UpdateWindowsAmi	使用此文档自动执行 image-maintenance 任务。有关更多信息，请参阅 演练：使用 AWS-UpdateWindowsAmi 自定义和更新 Windows AMI (p. 122)。
AWSSupport-ExecuteEC2Rescue	使用此文档诊断并解决有关 Amazon EC2 实例的问题。有关更多信息，请参阅 在无法访问的实例上运行 EC2Rescue 工具 (p. 133)。
AWSSupport-ResetAccess	使用此文档在 Amazon EC2 Windows 实例上自动重新启用本地管理员密码生成。有关更多信息，请参阅 在 Amazon EC2 实例上重置密码和 SSH 密钥 (p. 137)。

主题

- [演练：使用 AWS-UpdateLinuxAmi 自定义和更新 Linux AMI](#) (p. 120)
- [演练：使用 AWS-UpdateWindowsAmi 自定义和更新 Windows AMI](#) (p. 122)

演练：使用 AWS-UpdateLinuxAmi 自定义和更新 Linux AMI

使用 AWS-UpdateLinuxAmi 文档可以自动完成映像维护任务，而无需使用 JSON 或 YAML 编写工作流程。您可以使用 AWS-UpdateLinuxAmi 文档执行以下类型的任务。

- 在 Amazon Linux、Red Hat、Ubuntu、SLES 或 Cent OS Amazon Machine Image (AMI) 上升级所有分发程序包和 Amazon 软件。这是默认文档行为。
- 在现有映像上安装 SSM 代理以启用 Systems Manager 功能，例如使用 Run Command 的远程命令执行，或者使用 Inventory 的软件清单收集。
- 安装其他软件包。

开始前的准备工作

在您开始使用 Automation 文档之前，请先为 Automation 配置角色和 CloudWatch Events (后者可选)。有关更多信息，请参阅 [设置 Automation \(p. 107\)](#)。

AWS-UpdateLinuxAmi 文档接受以下输入参数。

参数	类型	描述
SourceAmiId	字符串	(必需) 源 AMI ID。
InstanceIamRole	字符串	(可选) 您在 设置 Automation (p. 107) 中创建的 AWS Identity and Access Management (IAM) 实例配置文件角色的名称。实例配置文件角色为 Automation 提供在您实例上执行操作的权限，例如执行命令或启动和停止服务。Automation 文档仅使用实例配置文件角色的名称。如果您指定 Amazon 资源名称 (ARN)，则 Automation 执行失败。
AutomationAssumeRole	字符串	(可选) 您在 设置 Automation (p. 107) 中创建的 IAM 服务角色的名称。服务角色 (也称为代入角色) 为 Automation 提供权限，用于代入您的 IAM 角色和代表您执行操作。例如，在 Automation 文档中执行 <code>aws:createImage</code> 操作时，服务角色允许 Automation 创建新的 AMI。对于此参数，必须指定完整的 ARN。
TargetAmiName	字符串	(可选) 新 AMI 在创建之后的名称。默认名称是系统生成的字符串，其中包括源 AMI ID 以及创建时间和日期。
InstanceType	字符串	(可选) 启动作为工作区主机的实例的类型。实例类型因区域而异。默认类型为 <code>t2.micro</code> 。
PreUpdateScript	字符串	(可选) 在应用更新前要运行的脚本的 URL。默认值 (<code>\\"none\\"</code>) 不运行脚本。
PostUpdateScript	字符串	(可选) 在应用程序包更新后要运行的脚本的 URL。默认值 (<code>\\"none\\"</code>) 不运行脚本。
IncludePackages	字符串	(可选) 仅更新这些指定的程序包。默认值 (<code>\\"all\\"</code>) 将应用所有可用的更新。
ExcludePackages	字符串	(可选) 在所有情况下从更新中排除的程序包的名称。默认值 (<code>\\"none\\"</code>) 不排除任何程序包。

Automation 步骤

默认情况下 AWS-UpdateLinuxAmi 文档包括以下 Automation 步骤。

步骤 1：launchInstance (aws:runInstances 操作)

此步骤使用 Amazon EC2 用户数据和 IAM 实例配置文件角色启动实例。用户数据根据操作系统安装相应的 SSM 代理。安装 SSM 代理后，您可以利用 Systems Manager 功能，例如 Run Command、State Manager 和 Inventory。

步骤 2：updateOSSoftware (aws:runCommand 操作)

此步骤在已启动实例上运行以下命令：

- 从 Amazon S3 下载更新脚本。
- 运行可选的更新前脚本。
- 更新分发程序包和 Amazon 软件。
- 运行可选的更新后脚本。

执行日志存储在 /tmp 文件夹中，供用户以后查看。

如果您希望升级特定程序包集，则可以使用 `IncludePackages` 参数提供列表。在提供时，系统仅尝试更新这些程序包及其依赖项。不执行任何其他更新。默认情况下，如果未指定包含 程序包，则程序将更新所有可用程序包。

如果要在升级中排除特定程序包集，则可以向 `ExcludePackages` 参数提供列表。如果提供，这些程序包保持其当前版本，与任何其他指定的选项无关。默认情况下，在未指定任何排除 程序包时，将不排除程序包。

步骤 3：stopInstance (aws:changeInstanceState 操作)

此步骤停止已更新实例。

步骤 4：createImage (aws:createImage 操作)

此步骤创建一个新 AMI，带有可将其链接到源 ID 和创建时间的描述性名称。例如：“EC2 Automation 在 {{global:DATE_TIME}} 从 {{SourceAmiId}} 生成了 AMI”，其中 DATE_TIME 和 SourceID 表示 Automation 变量。

步骤 5: terminateInstance (aws:changeInstanceState 操作)

此步骤通过终止正在运行的实例来清除执行过程。

输出

执行过程返回新的 AMI ID 作为输出。

您可以使用 AWS-UpdateLinuxAmi 文档作为模板来创建您自己的文档，如接下来的部分中所述。有关 Automation 文档中支持的操作 (步骤) 的信息，请参阅 [Systems Manager 自动化文档参考 \(p. 334\)](#)。有关如何使用 Automation 文档的信息，请参阅 [Systems Manager Automation 演练 \(p. 114\)](#)。

演练：使用 AWS-UpdateWindowsAmi 自定义和更新 Windows AMI

使用 AWS-UpdateWindowsAmi 文档可以在 Amazon Windows AMI 上自动完成映像维护任务，而无需使用 JSON 或 YAML 编写工作流程。本文档在 Windows Server 2008 R2 或更高版本中受支持。您可以使用 AWS-UpdateWindowsAmi 文档执行以下类型的任务。

- 安装所有 Windows 更新并升级 Amazon 软件 (默认行为)。
- 安装特定 Windows 更新并升级 Amazon 软件。

- 使用您的脚本自定义 AMI。

开始前的准备工作

在您开始使用 Automation 文档之前，请先为 Automation 配置角色和 CloudWatch Events (后者可选)。有关更多信息，请参阅 [设置 Automation \(p. 107\)](#)。

Note

SSM 代理的更新通常会在不同时间向不同区域推广。当您自定义或更新 AMI 时，请仅使用为您工作所在的区域发布的源 AMI。这将确保您使用该区域发布的最新 SSM 代理并避免兼容性问题。

AWS-UpdateWindowsAmi 文档接受以下输入参数。

参数	类型	描述
SourceAmiId	字符串	(必需) 源 AMI ID。
InstanceIamRole	字符串	(可选) 您在 设置 Automation (p. 107) 中创建的 AWS Identity and Access Management (IAM) 实例配置文件角色的名称。实例配置文件角色为 Automation 提供在您实例上执行操作的权限，例如执行命令或启动和停止服务。Automation 文档仅使用实例配置文件角色的名称。如果您指定 Amazon 资源名称 (ARN)，则 Automation 执行失败。
AutomationAssumeRole	字符串	(可选) 您在 设置 Automation (p. 107) 中创建的 IAM 服务角色的名称。服务角色 (也称为代入角色) 为 Automation 提供权限，用于代入您的 IAM 角色和代表您执行操作。例如，在 Automation 文档中执行 <code>aws:createImage</code> 操作时，服务角色允许 Automation 创建新的 AMI。对于此参数，必须指定完整的 ARN。
TargetAmiName	字符串	(可选) 新 AMI 在创建之后的名称。默认名称是系统生成的字符串，其中包括源 AMI ID 以及创建时间和日期。
InstanceType	字符串	(可选) 启动作为工作区主机的实例的类型。实例类型因区域而异。默认类型为 <code>t2.medium</code> 。
PreUpdateScript	字符串	(可选) 要在更新 AMI 之前运行的脚本。在 Automation 文档中或在运行时输入一个脚本作为参数。
PostUpdateScript	字符串	(可选) 要在更新 AMI 之后运行的脚本。在 Automation 文档中或在运行时输入一个脚本作为参数。

参数	类型	描述
IncludeKbs	字符串	(可选) 指定一个或多个要包括的 Microsoft 知识库 (KB) 文章 ID。可以使用逗号分隔值安装多个 ID。有效格式：KB9876543 或 9876543。
ExcludeKbs	字符串	(可选) 指定一个或多个要排除的 Microsoft 知识库 (KB) 文章 ID。可以使用逗号分隔值排除多个 ID。有效格式：KB9876543 或 9876543。
类别	字符串	(可选) 指定一个或多个更新类别。可以使用逗号分隔值筛选类别。选项：关键更新、安全更新、定义更新、Update Rollup、Service Pack、工具、更新或驱动程序。有效格式包括单个条目，例如：关键更新。或者，可以指定逗号分隔列表：关键更新,安全更新,定义更新。
SeverityLevels	字符串	(可选) 指定一个或多个与更新关联的 MSRC 严重性级别。可以使用逗号分隔值筛选严重性级别。选项：关键、重要、低、中或未指定。有效格式包括单个条目，例如：关键。或者，可以指定逗号分隔列表：关键,重要,低。

Automation 步骤

默认情况下 AWS-UpdateWindowsAmi 文档包括以下 Automation 步骤。

步骤 1：launchInstance (aws:runInstances 操作)

此步骤以从指定的 SourceAmiID 启动一个具有 IAM 实例配置文件角色的实例。

步骤 2：runPreUpdateScript (aws:runCommand 操作)

此步骤可让您以字符串形式指定一个在安装更新前运行的脚本。

步骤 3：updateEC2Config (aws:runCommand 操作)

此步骤使用 AWS-InstallPowerShellModule 公有文档下载 AWS 公有 PowerShell 模块。Systems Manager 使用 SHA-256 哈希验证模块的完整性。然后 Systems Manager 检查操作系统以确定是要更新 EC2Config 还是 EC2Launch。EC2Config 通过 Windows Server 2012 R2 在 Windows Server 2008 R2 上运行。EC2Launch 在 Windows Server 2016 上运行。

步骤 4：updateSSMAgent (aws:runCommand 操作)

此步骤使用 AWS-UpdateSSMAgent 公有文档更新 SSM 代理。

步骤 5：updateAWSPVDriver (aws:runCommand 操作)

此步骤使用 AWS-ConfigureAWSPackage 公有文档更新 AWS PV 驱动程序。

步骤 6：updateAwsEnaNetworkDriver (aws:runCommand 操作)

此步骤使用 AWS-ConfigureAWSPackage 公有文档更新 AWS ENA 网络驱动程序。

步骤 7 : installWindowsUpdates (aws:runCommand 操作)

此步骤使用 AWS-InstallWindowsUpdates 公有文档安装 Windows 更新。默认情况下, Systems Manager 搜索并安装任何缺失的更新。可以通过指定下列参数之一更改默认行为: IncludeKbs、ExcludeKbs、Categories 或 SeverityLevels。

步骤 8 : runPostUpdateScript (aws:runCommand 操作)

此步骤可让您以字符串形式指定一个在安装更新后运行的脚本。

步骤 9 : runSysprepGeneralize (aws:runCommand 操作)

此步骤使用 AWS-InstallPowerShellModule 公有文档下载 AWS 公有 PowerShell 模块。Systems Manager 使用 SHA-256 哈希验证模块的完整性。然后 Systems Manager 使用 AWS 支持的方法针对 EC2Launch (Windows Server 2016) 或 EC2Config (Windows Server 2008 R2 到 2012 R2) 运行 sysprep。

步骤 10 : stopInstance (aws:changeInstanceState 操作)

此步骤停止已更新实例。

步骤 11 : createImage (aws:createImage 操作)

此步骤创建一个新 AMI, 带有可将其链接到源 ID 和创建时间的描述性名称。例如: “EC2 Automation 在 {{global:DATE_TIME}} 从 {{SourceAmiId}} 生成了 AMI”, 其中 DATE_TIME 和 SourceId 表示 Automation 变量。

步骤 12 : TerminateInstance (aws:changeInstanceState 操作)

此步骤通过终止正在运行的实例来清除执行过程。

输出

此部分可让您将各个步骤的输出或任何参数的值指定为 Automation 输出。默认情况下, 输出是由执行创建的已更新 Windows AMI 的 ID。

您可以使用 AWS-UpdateWindowsAmi 文档作为模板来创建您自己的文档, 如接下来的部分中所述。有关 Automation 文档中支持的操作 (步骤) 的信息, 请参阅 [Systems Manager 自动化文档参考 \(p. 334\)](#)。有关如何使用 Automation 文档的信息, 请参阅 [Systems Manager Automation 演练 \(p. 114\)](#)。

创建动态自动化工作流程

默认情况下, 您在自动化文档的 mainSteps 部分中定义的步骤 (或操作) 将按先后顺序执行。在一个操作完成后, mainSteps 部分中指定的下一个操作将开始。此外, 如果一个操作无法执行, 则整个自动化工作流程将失败 (默认情况下)。您可以使用此部分中描述的选项来创建执行条件分支的自动化工作流程。这意味着, 您可以创建在步骤完成时动态响应条件更改的自动化工作流程。以下是可用于创建动态自动化工作流程的选项的列表。

- nextStep : 指定自动化工作流程中的哪个步骤在成功完成一个步骤后进行处理。
- isEnd : 在特定步骤结束时停止自动化执行。如果步骤执行失败, 自动化执行将停止, 否则将成功。该选项的默认值为 false。
- isCritical : 将一个步骤指定为成功完成自动化的关键步骤。如果具有此分派的步骤失败, 则自动化会将自动化的最终状态报告为失败。该选项的默认值为 true。
- onFailure : 指示工作流程在失败时是应中止、继续还是转到其他步骤。该选项的默认值为“中止”。

如何使用动态工作流程选项的示例

本部分包括有关如何使用自动化文档中的动态工作流程选项的其他示例。本部分中的每个示例均扩展以下自动化文档。此文档包含两个操作。第一个操作名为 InstallMsiPackage。它使用 aws:runCommand 操作将应

用程序安装到 Windows Server 实例上。第二个操作名为 TestInstall。它使用 aws:invokeLambdaFunction 操作在成功安装应用程序后测试该应用程序。步骤 1 指定 "onFailure":"Abort"。这意味着，如果应用程序未成功安装，则自动化工作流程执行会在进入步骤 2 前停止。

示例 1：带两个线性操作的自动化文档

```
{
  "schemaVersion":"0.3",
  "description":"Install MSI package and run validation.",
  "assumeRole":{"automationAssumeRole"}",
  "parameters":{
    "automationAssumeRole":{
      "type":"String",
      "description":"(Required) Assume role."
    },
    "packageName":{
      "type":"String",
      "description":"(Required) MSI package to be installed."
    },
    "instanceIds":{
      "type":"String",
      "description":"(Required) Comma separated list of instances."
    }
  }
  "mainSteps":[
    {
      "name":"InstallMsiPackage",
      "action":"aws:runCommand",
      "maxAttempts":2,
      "onFailure":"Abort",
      "inputs":{
        "InstanceIds":[
          {
            {
              instanceIds
            }
          ]
        },
        "DocumentName":"AWS-RunPowerShellScript",
        "Parameters":{
          "commands":[
            "msiexec /i {{packageName}}"
          ]
        }
      }
    },
    {
      "name":"TestInstall",
      "action":"aws:invokeLambdaFunction",
      "maxAttempts":1,
      "timeoutSeconds":500,
      "inputs":{
        "FunctionName":"TestLambdaFunction"
      }
    }
  ]
}
```

创建跳转到不同步骤的动态工作流程

以下示例使用 "onFailure":"step:**step_name**"、nextStep 和 isEnd 选项来创建动态自动化工作流程。在此示例中，如果 InstallMsiPackage 操作失败，则工作流程将跳转至名为 PostFailure ("onFailure":"step:PostFailure") 的操作以运行 AWS Lambda 函数，从而在安装失败时执行某个操作。如果安装成功，则工作流程进程将跳转至 TestInstall 操作 ("nextStep":"TestInstall")。TestInstall 和 PostFailure 步骤都使用 isEnd 选项 ("isEnd":"true")，以便工作流程在任一步骤完成时完成工作流程执行。

Note

可以选择在 mainSteps 操作的最后一步中使用 isEnd 选项。如果最后一个步骤未跳转到其他步骤，则自动化工作流程会在最后一步中执行操作后停止。

示例 2：跳转到不同步骤的动态工作流程

```
"mainSteps":[
  {
    "name":"InstallMsiPackage",
    "action":"aws:runCommand",
    "onFailure":"step:PostFailure",

    "maxAttempts":2,
    "inputs":{"
      "InstanceIds":[
        {
          {
            "i-1234567890abcdef0,i-0598c7d356eba48d7"
          }
        }
      ],
      "DocumentName":"AWS-RunPowerShellScript",
      "Parameters":{"
        "commands":[
          "msiexec /i {{packageName}}"
        ]
      }
    }
  },
  "nextStep":"TestInstall"
},
{
  "name":"TestInstall",
  "action":"aws:invokeLambdaFunction",
  "maxAttempts":1,
  "timeoutSeconds":500,
  "inputs":{"
    "FunctionName":"TestLambdaFunction"
  },
  "isEnd":"true"
},
{
  "name":"PostFailure",
  "action":"aws:invokeLambdaFunction",
  "maxAttempts":1,
  "timeoutSeconds":500,
  "inputs":{"
    "FunctionName":"PostFailureRecoveryLambdaFunction"
  },
  "isEnd":"true"
}
]
```

Note

在处理自动化文档时，系统将确认文档不会创建无限循环。如果检测到无限循环，则自动化将返回一个错误和一个显示哪些步骤创建循环的循环跟踪。

创建定义关键步骤的动态工作流程

您可以指定一个对于自动化工作流程的整体成功很重要的步骤。如果关键步骤失败，则自动化会将执行状态报告为失败，即使已成功执行一个或多个步骤也是如此。在以下示例中，用户在 InstallMsiPackage 步骤失败 ("onFailure":"step:VerifyDependencies") 时标识 VerifyDependencies 步骤。用户指定 InstallMsiPackage 步骤不是关键步骤 ("isCritical":false)。在此示例中，如果应用程序安装失败，自动化将处理 VerifyDependencies 步骤以确定是否因一个或多个依赖项缺失而导致应用程序安装失败。

示例 3：定义自动化工作流程的关键步骤

```
{
  "name": "InstallMsiPackage",
  "action": "aws:runCommand",
  "onFailure": "step:VerifyDependencies",
  "isCritical": false,
  "maxAttempts": 2,
  "inputs": {
    "InstanceIds": [
      {
        {
          instanceIds
        }
      ]
    },
    "DocumentName": "AWS-RunPowerShellScript",
    "Parameters": {
      "commands": [
        "msiexec /i {{packageName}}"
      ]
    }
  },
  "nextStep": "TestPackage"
}
```

演练：创建一个 Automation 文档

本演练将介绍如何创建和运行自定义 Automation 文档。在您运行 Automation 后，系统会执行以下任务。

- 从指定的 AMI 启动 Windows 实例。
- 使用 Run Command 运行将 Windows 更新应用于实例的命令。
- 停止实例。
- 创建新的 Windows AMI。
- 为 Windows AMI 添加标签。
- 终止原始实例。

Automation 示例文档

自动化运行以 JSON 或 YAML 格式编写的 Systems Manager 自动化文档。自动化文档包括在工作流执行期间要执行的操作。有关 Systems Manager 文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。有关您可以添加到文档的操作的信息，请参阅 [Systems Manager 自动化文档参考 \(p. 334\)](#)。

Note

默认情况下，当 Automation 运行 AWS-UpdateWindowsAmi 文档并创建一个临时实例时，系统会使用默认 VPC (172.30.0.0/16)。如果您删除了默认 VPC，会收到以下错误：

VPC not defined 400

要解决此问题，您必须复制 AWS-UpdateWindowsAmi 文档并指定子网 ID。有关更多信息，请参阅 [VPC not defined 400 \(p. 161\)](#)。

使用 Automation 创建经过修补的 AMI

1. 收集以下信息。您将在此过程的稍后部分指定此信息。
 - 要更新的 AMI 的源 ID。
 - 创建 AWS Identity and Access Management (IAM) 实例配置文件角色和 Automation 服务角色 (或代入角色)。有关这些角色以及如何从 AWS CloudFormation 模板快速创建角色的更多信息，请参阅

阅 [方法 1：使用 AWS CloudFormation 为 Automation 配置角色 \(p. 108\)](#)。请确保复制实例配置文件角色的名称和 Automation 服务角色的 Amazon 资源名称 (ARN)，如 [复制 Automation 的角色信息 \(p. 109\)](#) 中所述。

2. 将以下示例文档复制到一个文本编辑器 (如 Notepad) 中。将 `assumeRole` 的值和 `IamInstanceProfileName` 的值分别更改为您之前为 Automation 创建 IAM 角色时所创建的角色 ARN 和角色名称。将该文档以 `patchWindowsAmi.json` 或 `patchWindowsAmi.yaml` 格式保存在本地硬盘上。

```
{
  "description": "Systems Manager Automation Demo - Patch and Create a New AMI",
  "schemaVersion": "0.3",
  "assumeRole": "the role ARN you created",
  "parameters": {
    "sourceAMIid": {
      "type": "String",
      "description": "AMI to patch"
    },
    "targetAMIname": {
      "type": "String",
      "description": "Name of new AMI",
      "default": "patchedAMI-{{global:DATE_TIME}}"
    }
  },
  "mainSteps": [
    {
      "name": "startInstances",
      "action": "aws:runInstances",
      "timeoutSeconds": 1200,
      "maxAttempts": 1,
      "onFailure": "Abort",
      "inputs": {
        "ImageId": "{{ sourceAMIid }}",
        "InstanceType": "m3.large",
        "MinInstanceCount": 1,
        "MaxInstanceCount": 1,
        "IamInstanceProfileName": "the name of the IAM role you created"
      }
    },
    {
      "name": "installMissingWindowsUpdates",
      "action": "aws:runCommand",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "DocumentName": "AWS-InstallMissingWindowsUpdates",
        "InstanceIds": [
          "{{ startInstances.InstanceIds }}"
        ],
        "Parameters": {
          "UpdateLevel": "Important"
        }
      }
    },
    {
      "name": "stopInstance",
      "action": "aws:changeInstanceState",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "InstanceIds": [
          "{{ startInstances.InstanceIds }}"
        ],
        "DesiredState": "stopped"
      }
    }
  ]
}
```

```

    },
    {
      "name": "createImage",
      "action": "aws:createImage",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "InstanceId": "{{ startInstances.InstanceIds }}",
        "ImageName": "{{ targetAMIname }}",
        "NoReboot": true,
        "ImageDescription": "AMI created by EC2 Automation"
      }
    },
    {
      "name": "createTags",
      "action": "aws:createTags",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "ResourceType": "EC2",
        "ResourceIds": [
          "{{ createImage.ImageId }}"
        ],
        "Tags": [
          {
            "Key": "Generated By Automation",
            "Value": "{{ automation:EXECUTION_ID }}"
          },
          {
            "Key": "From Source AMI",
            "Value": "{{ sourceAMId }}"
          }
        ]
      }
    },
    {
      "name": "terminateInstance",
      "action": "aws:changeInstanceState",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "InstanceIds": [
          "{{ startInstances.InstanceIds }}"
        ],
        "DesiredState": "terminated"
      }
    }
  ],
  "outputs": [
    "createImage.ImageId"
  ]
}

```

3. 将 AWS CLI [下载](#)到本地计算机上。
4. 编辑以下命令，并指定 patchWindowsAmi.json/yaml 文件在本地计算机上的路径。运行该命令，以创建必需的自动化文档。

Note

对于 name 参数，您不能使用 AWS 作为文档前缀。如果您指定 AWS-## 或 AWS##，则将收到错误。

```
aws ssm create-document --name "patchWindowsAmi" --content file:///Users/test-user/Documents/patchWindowsAmi.json/yaml --document-type Automation
```

系统将返回有关命令进程的信息。

```
{
  "DocumentDescription": {
    "Status": "Creating",
    "Hash": "bce98f80b89668b092cd094d2f2895f57e40942bcc1598d85338dc9516b0b7f1",
    "Name": "test",
    "Parameters": [
      {
        "Type": "String",
        "Name": "sourceAMIId",
        "Description": "AMI to patch"
      },
      {
        "DefaultValue": "patchedAMI-{{global:DATE_TIME}}",
        "Type": "String",
        "Name": "targetAMIname",
        "Description": "Name of new AMI"
      }
    ],
    "DocumentType": "Automation",
    "PlatformTypes": [
      "Windows",
      "Linux"
    ],
    "DocumentVersion": "1",
    "HashType": "Sha256",
    "CreateDate": 1488303738.572,
    "Owner": "12345678901",
    "SchemaVersion": "0.3",
    "DefaultVersion": "1",
    "LatestVersion": "1",
    "Description": "Systems Manager Automation Demo - Patch and Create a New AMI"
  }
}
```

5. 运行以下命令以查看您可以访问的文件列表。

```
aws ssm list-documents --document-filter-list key=Owner,value=Self
```

系统将返回类似于以下内容的信息：

```
{
  "DocumentIdentifiers":[
    {
      "Name":" patchWindowsAmi",
      "PlatformTypes": [

      ],
      "DocumentVersion": "5",
      "DocumentType": "Automation",
      "Owner": "12345678901",
      "SchemaVersion": "0.3"
    }
  ]
}
```

6. 运行以下命令以查看有关 patchWindowsAmi 文档的详细信息。

```
aws ssm describe-document --name patchWindowsAmi
```


系统将返回类似于以下内容的信息：

```
{
  "Document": {
    "Status": "Active",
    "Hash": "99d5b2e33571a6bb52c629283bca0a164026cd201876adf0a76de16766fb98ac",
    "Name": "patchWindowsAmi",
    "Parameters": [
      {
        "DefaultValue": "ami-3f0c4628",
        "Type": "String",
        "Name": "sourceAMIid",
        "Description": "AMI to patch"
      },
      {
        "DefaultValue": "patchedAMI-{{global:DATE_TIME}}",
        "Type": "String",
        "Name": "targetAMIname",
        "Description": "Name of new AMI"
      }
    ],
    "DocumentType": "Automation",
    "PlatformTypes": [

  ],
  "DocumentVersion": "5",
  "HashType": "Sha256",
  "CreateDate": 1478904417.477,
  "Owner": "12345678901",
  "SchemaVersion": "0.3",
  "DefaultVersion": "5",
  "LatestVersion": "5",
  "Description": "Automation Demo - Patch and Create a New AMI"
}
}
```

7. 运行以下命令以运行 patchWindowsAmi 文档和自动化工作流程。该命令采用两个输入参数：要修补的 AMI 的 ID 和新 AMI 的名称。以下示例命令使用的是最新的 EC2 AMI，以最大限度减少需应用的补丁数量。如果您多次运行此命令，则必须为 targetAMIname 指定唯一的值。AMI 名称必须是独一无二的。

```
aws ssm start-automation-execution --document-name="patchWindowsAmi" --parameters
sourceAMIid="ami-bd3ba0aa"
```

该命令将会返回执行 ID。请将该 ID 复制到剪贴板。您将使用该 ID 查看工作流的状态。

```
{
  "AutomationExecutionId": "ID"
}
```

您可以在 控制台中 监控工作流的状态。请检查控制台以确认是否正在启动新实例。实例启动完成后，您可以确认已运行 Run Command 操作。Run Command 执行完成之后，您应该会在 AMI 映像列表中看到新的 AMI。

8. 要使用 CLI 查看工作流程的执行情况，请运行以下命令：

```
aws ssm describe-automation-executions
```

9. 要查看有关执行进度的详细信息，请运行以下命令。

```
aws ssm get-automation-execution --automation-execution-id ID
```

Note

根据应用的补丁数量，在该示例工作流中运行的 Windows 修补过程可能需要 30 分钟或更长时间才能完成。

有关如何使用 Automation 的更多示例，包括在您刚完成的演练中构建的示例，请参阅 [Systems Manager Automation 示例 \(p. 133\)](#)。

Systems Manager Automation 示例

以下示例演示如何使用 Systems Manager Automation 来简化常见的示例和系统维护任务。请注意，其中一些示例在如何更新 Windows AMI 示例的基础之上扩展，该示例在 [演练：创建一个 Automation 文档 \(p. 128\)](#) 中介绍。

示例

- [在无法访问的实例上运行 EC2Rescue 工具 \(p. 133\)](#)
- [在 Amazon EC2 实例上重置密码和 SSH 密钥 \(p. 137\)](#)
- [使用 Automation、Lambda 和 Parameter Store 简化 AMI 修补 \(p. 141\)](#)
- [将 Automation 用于 Jenkins \(p. 146\)](#)
- [修补 AMI 和更新 Auto Scaling 组 \(p. 148\)](#)

在无法访问的实例上运行 EC2Rescue 工具

EC2Rescue 可以帮助您诊断并解决有关 Amazon EC2 Linux 和 Windows Server 实例的问题。您可以手动运行工具，如[使用适用于 Linux Server 的 EC2Rescue](#)和[使用适用于 Windows Server 的 EC2Rescue](#)。或者，您可以使用 Systems Manager Automation 和 AWSSupport-ExecuteEC2Rescue 文档自动运行工具。AWSSupport-ExecuteEC2Rescue 文档旨在执行 Systems Manager 操作、AWS CloudFormation 操作和 Lambda 函数的组合，从而将使用 EC2Rescue 通常所需的步骤自动化。

您可以使用 AWSSupport-ExecuteEC2Rescue 文档，对不同类型的操作系统 (OS) 问题进行故障排除和潜在修复。有关完整列表，请参阅以下主题：

Windows：请参阅[将适用于 Windows Server 的 EC2Rescue 与命令行配合使用](#)中的抢救操作。

Linux：一些适用于 Linux 的 EC2Rescue 模块会检测并尝试修复问题。有关更多信息，请参阅 GitHub 上各个模块的 [aws-ec2rescue-linux](#) 文档。

如何使用

使用自动化和 AWSSupport-ExecuteEC2Rescue 文档对实例进行故障排除的工作原理如下：

- 您为无法访问的实例指定 ID 并运行自动化工作流程。
- 系统创建一个临时 VPC，然后运行一系列 Lambda 函数以配置该 VPC。
- 系统在与您的原始实例相同的可用区内为您的临时 VPC 标识一个子网。
- 系统启动一个临时的启用了 SSM 的帮助程序实例。
- 系统停止您的原始实例并创建备份。然后，它将原始根卷挂载到帮助程序实例。
- 系统使用 Run Command 在帮助程序实例上运行 EC2Rescue。EC2Rescue 识别并尝试修复有关已挂载的原始根卷的问题。完成后，EC2Rescue 重新将根卷挂载回原始实例。
- 系统重启您的原始实例，并终止临时实例。系统也将终止临时 VPC 和在自动化开始时创建的 Lambda 函数。

开始前的准备工作

运行以下自动化之前，请执行以下操作：

- 复制无法访问的实例的实例 ID。您将在过程中指定此 ID。
- (可选) 收集无法访问实例所在可用区中子网的 ID。EC2Rescue 实例将在此子网中创建。如果不指定子网，Automation 会在您的 AWS 账户中创建新建一个临时 VPC。验证您的 AWS 账户是否至少有一个可用 VPC。默认情况下，一个区域中可以创建 5 个 VPC。如果您已在该区域中创建 5 个 VPC，则自动化将会失败，但不会对您的实例进行更改。有关更多信息，请参阅 [VPC 和子网](#)。
- 或者，您可以为自动化创建并指定一个 AWS Identity and Access Management (IAM) 角色。如果您不指定此角色，则自动化将在运行自动化的用户的环境中运行。有关为自动化创建角色的更多信息，请参阅 [快速入门第 2 步：通过使用 IAM 服务角色运行 Automation 工作流程 \(p. 104\)](#)。

向 AWSSupport-EC2Rescue 授予在您的实例上执行操作的权限

在自动化执行期间，EC2Rescue 需要权限才能在您的实例上执行一系列操作。这些操作调用 AWS Lambda、IAM 和 Amazon EC2 服务，安全地尝试修复您的实例的问题。如果您在 AWS 账户和/或 VPC 中具有管理员级权限，您可能能够在不按照本部分中所述配置权限的情况下运行自动化。如果您没有管理员级权限，则您或管理员必须使用以下选项之一配置权限。

- [使用 IAM 策略授予权限 \(p. 134\)](#)
- [使用 AWS CloudFormation 模板授予权限 \(p. 135\)](#)

使用 IAM 策略授予权限

您可以将以下 IAM 策略作为内链策略附加到您的 IAM 用户账户、组或角色；也可以创建一个新的 IAM 托管策略，并将其附加到您的用户账户、组或角色。有关将内联策略添加到用户账户、组或角色的更多信息，请参阅 [使用内联策略](#)。有关创建新的托管策略的更多信息，请参阅 [使用托管策略](#)。

Note

如果创建新的 IAM 托管策略，则还必须将 AmazonSSMAutomationRole 托管策略附加到该策略，以便实例与 Systems Manager API 通信。

适用于 AWSSupport-EC2Rescue 的 IAM 策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lambda:InvokeFunction",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource": "arn:aws:lambda:*:An-AWS-Account-ID:function:AWSSupport-EC2Rescue-*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::awssupport-ssm.*/*.template",
        "arn:aws:s3:::awssupport-ssm.*/*.zip"
      ],
      "Effect": "Allow"
    }
  ]
}
```

```

{
  "Action": [
    "iam:CreateRole",
    "iam:CreateInstanceProfile",
    "iam:GetRole",
    "iam:GetInstanceProfile",
    "iam:PutRolePolicy",
    "iam:DetachRolePolicy",
    "iam:AttachRolePolicy",
    "iam:PassRole",
    "iam:AddRoleToInstanceProfile",
    "iam:RemoveRoleFromInstanceProfile",
    "iam>DeleteRole",
    "iam>DeleteRolePolicy",
    "iam>DeleteInstanceProfile"
  ],
  "Resource": [
    "arn:aws:iam::An-AWS-Account-ID:role/AWSSupport-EC2Rescue-*",
    "arn:aws:iam::An-AWS-Account-ID:instance-profile/AWSSupport-EC2Rescue-*"
  ],
  "Effect": "Allow"
},
{
  "Action": [
    "lambda:CreateFunction",
    "ec2:CreateVpc",
    "ec2:ModifyVpcAttribute",
    "ec2>DeleteVpc",
    "ec2:CreateInternetGateway",
    "ec2:AttachInternetGateway",
    "ec2:DetachInternetGateway",
    "ec2>DeleteInternetGateway",
    "ec2:CreateSubnet",
    "ec2>DeleteSubnet",
    "ec2:CreateRoute",
    "ec2>DeleteRoute",
    "ec2:CreateRouteTable",
    "ec2:AssociateRouteTable",
    "ec2:DisassociateRouteTable",
    "ec2>DeleteRouteTable",
    "ec2:CreateVpcEndpoint",
    "ec2>DeleteVpcEndpoints",
    "ec2:ModifyVpcEndpoint",
    "ec2:Describe*"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
]
}

```

使用 AWS CloudFormation 模板授予权限

AWS CloudFormation 使用预配置模板自动化创建 IAM 角色和策略的过程。使用 AWS CloudFormation，通过以下过程为 EC2Rescue Automation 创建所需的 IAM 角色和策略。

为 EC2Rescue 创建所需的 IAM 角色和策略

1. 选择 Launch Stack 按钮。该按钮会打开 AWS CloudFormation 控制台并用 EC2Rescue 模板的 URL 填充 Specify an Amazon S3 template URL 字段。

Note

选择 View 以查看模板。

查看	启动
查看	

2. 选择 Next。
3. 在 Specify Details 页面上的 Stack Name 字段中，选择保留默认值或指定您自己的值。选择 Next。
4. 在 Options 页面上，您无需进行任何选择。选择 Next。
5. 在 Review 页面上，向下滚动并选择 I acknowledge that AWS CloudFormation might create IAM resources 选项。
6. 选择 Create。

大约三分分钟左右，AWS CloudFormation 会显示 CREATE_IN_PROGRESS 状态。创建堆栈后，状态将变为 CREATE_COMPLETE。

7. 在堆栈列表中，选择您刚刚创建的堆栈旁边的选项，然后选择 Outputs 选项卡。
8. 复制 Value。这是 AssumeRole 的 ARN。在运行自动化时，您将指定此 ARN。

执行自动化

Note

以下过程描述了您在 Amazon EC2 控制中执行的步骤。您也可以在新的 [AWS Systems Manager 控制台](#) 中执行这些步骤。新控制台中的步骤将不同于下面的步骤。

Important

以下自动化执行将停止无法访问的实例。停止实例可能导致挂载的实例存储卷 (如果存在) 上的数据丢失。如果未关联弹性 IP，停止实例也可能导致公有 IP 更改。

运行 AWSSupport-ExecuteEC2Rescue 自动化 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Automation。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Automation。

3. 选择执行自动化。
4. 在自动化文档部分中，选择列表中的我或 Amazon 所拥有。
5. 在文档列表中，选择 AWSSupport-ExecuteEC2Rescue。文档所有者是 Amazon。
6. 在文档详细信息部分中，验证文档版本是否设置为最高默认版本。例如，6 (默认)。
7. 在执行模式部分中，选择一次执行整个自动化。
8. 保留目标和速率控制选项为禁用状态。
9. 在输入参数部分中，指定以下参数：
 - a. 对于 UnreachableInstanceId，指定无法访问实例的 ID。
 - b. 如果在排查实例故障时需要收集操作系统级别的日志，请为 LogDestination 指定 Amazon S3 存储桶。日志会自动上传到此指定存储桶。
 - c. 对于 EC2RescueInstanceType，为 EC2Rescue 实例指定实例类型。默认实例类型为 t2.small。
 - d. 对于 SubnetId，指定无法访问实例所在可用区中某个现有 VPC 的子网。默认情况下，Systems Manager 会新建一个 VPC，但您可以根据需要指定现有 VPC 中的某个子网。

Note

如果未看到指定存储桶或子网 ID 的选项，请确认使用的是不是文档的最新默认版本。

- e. 对于 AssumeRole，如果是使用本主题前面介绍的 CloudFormation 过程为此自动化创建的角色，则指定从 CloudFormation 控制台复制的 AssumeRole ARN。

10. 选择执行自动化。

作为工作流程的一部分，自动化将创建备份 AMI。自动化工作流程创建的所有其他资源都会自动删除，但此 AMI 将保留在您的账户中。此 AMI 遵从以下命名约定：

备份 AMI：AWSSupport-EC2Rescue:*UnreachableInstanceId*

在 Amazon EC2 控制台中，可以通过搜索自动化执行 ID 查找此 AMI。

在 Amazon EC2 实例上重置密码和 SSH 密钥

您可以使用 AWSSupport-ResetAccess 文档在 Amazon EC2 Windows 实例上自动重新启用本地管理员密码生成，以及在 Amazon EC2 Linux 实例上生成新 SSH 密钥。AWSSupport-ResetAccess 文档旨在执行 Systems Manager 操作、AWS CloudFormation 操作和 Lambda 函数的组合，从而将重置本地管理员密码通常所需的步骤自动化。

您可以使用 Automation 配合 AWSSupport-ResetAccess 文档解决以下问题：

Windows

您丢失了 EC2 密钥对：要解决此问题，您可以使用 AWSSupport-ResetAccess 文档，从当前实例创建启用了密码的 AMI，从 AMI 启动新实例，然后选择您拥有的密钥对。

您丢失了本地管理员密码：要解决此问题，您可以使用 AWSSupport-ResetAccess 文档生成可以使用当前 EC2 密钥对解密的一个新密码。

Linux

您丢失了 EC2 密钥对，或者配置了对实例的 SSH 访问但丢失：要解决此问题，您可以使用 AWSSupport-ResetAccess 文档创建当前实例的新 SSH 密钥，这使您能够重新连接到该实例。

Note

如果您的 EC2 Windows 实例为 Systems Manager 进行了配置，也可以使用 EC2Rescue 和 Run Command 重置您的本地管理员密码。有关更多信息，请参阅 Amazon EC2 User Guide for Windows Instances 中的[通过 Systems Manager Run Command 将 EC2Rescue 用于 Windows Server](#)。

如何使用

使用自动化和 AWSSupport-ResetAccess 文档对实例进行故障排除的工作原理如下：

- 您为实例指定 ID 并运行自动化工作流程。
- 系统创建一个临时 VPC，然后运行一系列 Lambda 函数以配置该 VPC。
- 系统在与您的原始实例相同的可用区内为您的临时 VPC 标识一个子网。
- 系统启动一个临时的启用了 SSM 的帮助程序实例。
- 系统停止您的原始实例并创建备份。然后，它将原始根卷挂载到帮助程序实例。
- 系统使用 Run Command 在帮助程序实例上运行 EC2Rescue。在 Windows 上，EC2Rescue 通过在附加的原始根卷上使用 EC2Config 或 EC2Launch 为本地管理员启用密码生成。在 Linux 上，EC2Rescue 生成并注入新的 SSH 密钥并将私有密钥加密保存到 Parameter Store 中。完成后，EC2Rescue 重新将根卷挂载回原始实例。
- 系统根据您的实例创建新 Amazon Machine Image (AMI)，现在密码生成已启用。您可以使用此 AMI 创建新 EC2 实例，并根据需要关联新密钥对。

- 系统重启您的原始实例，并终止临时实例。系统也将终止临时 VPC 和在自动化开始时创建的 Lambda 函数。
- Windows：您的实例生成一个新密码，您可以使用分配给实例的当前密钥对从 EC2 控制台对该密码进行解码。

Linux：您可以使用存储在 Systems Manager Parameter Store 中的 SSH 密钥 (格式为 /ec2rl/openssh/**instance_id**/key)，通过 SSH 连接到实例。

开始前的准备工作

运行以下自动化之前，请执行以下操作：

- 复制要重置管理员密码的实例的实例 ID。您将在过程中指定此 ID。
- (可选) 收集无法访问实例所在可用区中子网的 ID。EC2Rescue 实例将在此子网中创建。如果不指定子网，Automation 会在您的 AWS 账户中创建新建一个临时 VPC。验证您的 AWS 账户是否至少有一个可用 VPC。默认情况下，一个区域中可以创建 5 个 VPC。如果您已在该区域中创建 5 个 VPC，则自动化将会失败，但不会对您的实例进行更改。有关更多信息，请参阅 [VPC 和子网](#)。
- 或者，您可以为自动化创建并指定一个 AWS Identity and Access Management (IAM) 角色。如果您不指定此角色，则自动化将在运行自动化的用户的环境中运行。有关为自动化创建角色的更多信息，请参阅 [快速入门第 2 步：通过使用 IAM 服务角色运行 Automation 工作流程 \(p. 104\)](#)。

向 AWSSupport-EC2Rescue 授予在您的实例上执行操作的权限

在自动化执行期间，EC2Rescue 需要权限才能在您的实例上执行一系列操作。这些操作调用 AWS Lambda、IAM 和 Amazon EC2 服务，安全地尝试修复您的实例的问题。如果您在 AWS 账户和/或 VPC 中具有管理员级权限，您可能能够在不按照本部分中所述配置权限的情况下运行自动化。如果您没有管理员级权限，则您或管理员必须使用以下选项之一配置权限。

- [使用 IAM 策略授予权限 \(p. 138\)](#)
- [使用 AWS CloudFormation 模板授予权限 \(p. 135\)](#)

使用 IAM 策略授予权限

您可以将以下 IAM 策略作为内联策略附加到您的 IAM 用户账户、组或角色；也可以创建一个新的 IAM 托管策略，并将其附加到您的用户账户、组或角色。有关将内联策略添加到用户账户、组或角色的更多信息，请参阅 [使用内联策略](#)。有关创建新的托管策略的更多信息，请参阅 [使用托管策略](#)。

Note

如果创建新的 IAM 托管策略，则还必须将 AmazonSSMAutomationRole 托管策略附加到该策略，以便实例与 Systems Manager API 通信。

适用于 AWSSupport-ResetAccess 的 IAM 策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lambda:InvokeFunction",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource": "arn:aws:lambda:*:An-AWS-Account-ID:function:AWSSupport-EC2Rescue-*",
      "Effect": "Allow"
    }
  ],
  {
```



```

        "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
        ],
        "Resource": [
            "arn:aws:s3:::awssupport-ssm.*/*.template",
            "arn:aws:s3:::awssupport-ssm.*/*.zip"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "iam:CreateRole",
            "iam:CreateInstanceProfile",
            "iam:GetRole",
            "iam:GetInstanceProfile",
            "iam:PutRolePolicy",
            "iam:DetachRolePolicy",
            "iam:AttachRolePolicy",
            "iam:PassRole",
            "iam:AddRoleToInstanceProfile",
            "iam:RemoveRoleFromInstanceProfile",
            "iam>DeleteRole",
            "iam>DeleteRolePolicy",
            "iam>DeleteInstanceProfile"
        ],
        "Resource": [
            "arn:aws:iam::An-AWS-Account-ID:role/AWSSupport-EC2Rescue-*",
            "arn:aws:iam::An-AWS-Account-ID:instance-profile/AWSSupport-EC2Rescue-*"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "lambda:CreateFunction",
            "ec2:CreateVpc",
            "ec2:ModifyVpcAttribute",
            "ec2>DeleteVpc",
            "ec2:CreateInternetGateway",
            "ec2:AttachInternetGateway",
            "ec2:DetachInternetGateway",
            "ec2>DeleteInternetGateway",
            "ec2:CreateSubnet",
            "ec2>DeleteSubnet",
            "ec2:CreateRoute",
            "ec2>DeleteRoute",
            "ec2:CreateRouteTable",
            "ec2:AssociateRouteTable",
            "ec2:DisassociateRouteTable",
            "ec2>DeleteRouteTable",
            "ec2:CreateVpcEndpoint",
            "ec2>DeleteVpcEndpoints",
            "ec2:ModifyVpcEndpoint",
            "ec2:Describe*"
        ],
        "Resource": "*",
        "Effect": "Allow"
    }
]
}

```

使用 AWS CloudFormation 模板授予权限

AWS CloudFormation 使用预配置模板自动化创建 IAM 角色和策略的过程。使用 AWS CloudFormation，通过以下过程为 EC2Rescue Automation 创建所需的 IAM 角色和策略。

为 EC2Rescue 创建所需的 IAM 角色和策略

1. 选择 Launch Stack 按钮。该按钮会打开 AWS CloudFormation 控制台并用 EC2Rescue 模板的 URL 填充 Specify an Amazon S3 template URL 字段。

Note

选择 View 以查看模板。

查看	启动
查看	

2. 选择 Next。
3. 在 Specify Details 页面上的 Stack Name 字段中，选择保留默认值或指定您自己的值。选择 Next。
4. 在 Options 页面上，您无需进行任何选择。选择 Next。
5. 在 Review 页面上，向下滚动并选择 I acknowledge that AWS CloudFormation might create IAM resources 选项。
6. 选择 Create。

大约三分分钟左右，AWS CloudFormation 会显示 CREATE_IN_PROGRESS 状态。创建堆栈后，状态将变为 CREATE_COMPLETE。

7. 在堆栈列表中，选择您刚刚创建的堆栈旁边的选项，然后选择 Outputs 选项卡。
8. 复制 Value。这是 AssumeRole 的 ARN。在运行自动化时，您将指定此 ARN。

Note

此过程在 US East (Ohio) Region (us-east-2) 中创建一个 AWS CloudFormation 堆栈，但此过程创建的 IAM 角色为全局资源，可在所有区域中使用。

执行自动化

Note

以下过程描述了您在 Amazon EC2 控制台中执行的步骤。您也可以在新的 [AWS Systems Manager 控制台](#) 中执行这些步骤。新控制台中的步骤将不同于下面的步骤。

以下过程介绍如何使用 Amazon EC2 控制台运行 AWSSupport-ResetAccess 文档。

Important

以下自动化执行将停止实例。停止实例可能导致挂载的实例存储卷 (如果存在) 上的数据丢失。如果未关联弹性 IP，停止实例也可能导致公有 IP 更改。要避免这些配置更改，请使用 Run Command 重置访问权限。有关更多信息，请参阅 Amazon EC2 User Guide for Windows Instances 中的 [通过 Systems Manager Run Command 将 EC2Rescue 用于 Windows Server](#)。

运行 AWSSupport-ResetAccess 自动化 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Automation。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Automation。

3. 选择执行自动化。
4. 在 Document name 部分中，选择列表中的 Owned by Me or Amazon。
5. 在文档列表中，选择 AWSSupport-ResetAccess。文档所有者是 Amazon。
6. 在文档详细信息部分中，验证文档版本是否设置为最高默认版本。例如，4 (默认)。
7. 在执行模式部分中，选择一次执行整个自动化。
8. 保留目标和速率控制选项为禁用状态。
9. 在输入参数部分中，指定以下参数：
 - a. 对于 InstanceID，指定无法访问实例的 ID。
 - b. 对于 EC2RescueInstanceType，为 EC2Rescue 实例指定实例类型。默认实例类型为 t2.small。
 - c. 对于 SubnetId，指定您指定的实例所在可用区中某个现有 VPC 的子网。默认情况下，Systems Manager 会新建一个 VPC，但您可以根据需要指定现有 VPC 中的某个子网。

Note

如果未看到指定子网 ID 的选项，请确认使用的是不是文档的最新默认版本。

- d. 对于代入角色，如果是使用本主题前面介绍的 CloudFormation 过程为此自动化创建的角色，则指定从 CloudFormation 控制台复制的 AssumeRole ARN。
10. 选择执行自动化。
11. 要监控执行过程，请选择正在运行的自动化，然后选择 Steps 选项卡。完成执行后，选择 Descriptions 选项卡，然后选择 View output 以查看结果。要查看单个步骤的输出，请选择 Steps 选项卡，然后选择步骤旁的 View Outputs。

作为工作流程的一部分，自动化将创建备份 AMI 和启用了密码的 AMI。自动化工作流程创建的所有其他资源都会自动删除，但这些 AMI 将保留在您的账户中。这些 AMI 遵从以下命名约定：

- 备份 AMI：AWSSupport-EC2Rescue:*InstanceId*
- 启用了密码的 AMI：AWSSupport-EC2Rescue: Password-enabled AMI from *InstanceId*

可以通过搜索自动化执行 ID 查找这些 AMI。

对于 Linux，您的实例的新 SSH 私有密钥加密保存到 Parameter Store 中。参数名称为 /ec2r/openssh/*instance_id*/key。

使用 Automation、Lambda 和 Parameter Store 简化 AMI 修补

以下示例在如何更新 Windows AMI 基础之上扩展，这些内容在 [演练：创建一个 Automation 文档 \(p. 128\)](#) 中介绍。此示例使用组织维护和定期修补自己专有 AMI 的模型，而非构建自 Amazon EC2 AMI。

以下过程演示如何自动应用操作系统 (OS) 补丁到已视为最新的 Windows AMI 或最新 AMI。在示例中，参数 SourceAmiId 的默认值由名为 latestAmi 的 Systems Manager Parameter Store 参数定义。latestAmi 的值由 AWS Lambda 函数在 Automation 工作流结束时更新。由于此 Automation 过程，修补 AMI 所需的时间和工作量都尽可能减少，因为修补操作始终都应用到最新的 AMI。

开始前的准备工作

配置 Automation 角色以及 (可选) 为 Automation 配置 CloudWatch Events。有关更多信息，请参阅 [设置 Automation \(p. 107\)](#)。

内容

- [任务 1：在 Systems Manager Parameter Store 中创建一个参数 \(p. 142\)](#)
- [任务 2：为 AWS Lambda 创建 IAM 角色 \(p. 142\)](#)
- [步骤 3：创建 AWS Lambda 函数 \(p. 142\)](#)

- [任务 4：创建 Automation 文档并修补 AMI \(p. 144\)](#)

任务 1：在 Systems Manager Parameter Store 中创建一个参数

在 Parameter Store 中创建一个使用以下信息的字符串参数：

- 名称：latestAmi。
- 值：Windows AMI ID。例如：ami-188d6e0e。

有关如何创建 Parameter Store 字符串参数的信息，请参阅[创建 Systems Manager 参数 \(p. 369\)](#)。

任务 2：为 AWS Lambda 创建 IAM 角色

使用以下过程为 AWS Lambda 创建 IAM 服务角色。此角色包括 AWSLambdaExecute 和 AmazonSSMFullAccess 托管策略。这些策略授予 Lambda 权限，以便使用 Lambda 函数和 Systems Manager 来更新 latestAmi 参数的值。

为 Lambda 创建 IAM 服务角色

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中选择 Roles，然后选择 Create New Role。
3. 对于 Role name，键入可帮助您识别此角色用途的角色名称，例如，lambda-ssm-role。角色名称在您的 AWS 账户内必须是唯一的。键入名称后，选择页面底部的 Next Step。

Note

由于多个实体可能引用该角色，因此创建角色后无法更改角色的名称。

4. 在 Select Role Type 页面中，选择 AWS Service Roles 部分，然后选择 AWS Lambda。
5. 在 Attach Policy 页面中，选择 AWSLambdaExecute 和 AmazonSSMFullAccess，然后选择 Next Step。
6. 选择 Create Role。

步骤 3：创建 AWS Lambda 函数

使用以下过程创建 Lambda 函数，该函数会自动更新 latestAmi 参数的值。

创建 Lambda 函数

1. Sign in to the AWS Management Console and open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
2. 选择 Create a Lambda function。
3. 在 Select blueprint 页面上，选择 Blank Function。
4. 在 Configure triggers 页面上，选择 Next。
5. 在 Configure function 页面的 Name 字段中键入 Automation-UpdateSsmParam，如果需要还可键入说明。
6. 在 Runtime 列表中，选择 Python 2.7。
7. 在 Lambda function code 部分中，删除字段中预填充的代码，然后粘贴以下代码示例。

```
from __future__ import print_function

import json
import boto3

print('Loading function')
```

```
#Updates an SSM parameter
#Expects parameterName, parameterValue
def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    # get SSM client
    client = boto3.client('ssm')

    #confirm parameter exists before updating it
    response = client.describe_parameters(
        Filters=[
            {
                'Key': 'Name',
                'Values': [ event['parameterName'] ]
            },
        ]
    )

    if not response['Parameters']:
        print('No such parameter')
        return 'SSM parameter not found.'

    #if parameter has a Description field, update it PLUS the Value
    if 'Description' in response['Parameters'][0]:
        description = response['Parameters'][0]['Description']

        response = client.put_parameter(
            Name=event['parameterName'],
            Value=event['parameterValue'],
            Description=description,
            Type='String',
            Overwrite=True
        )

    #otherwise just update Value
    else:
        response = client.put_parameter(
            Name=event['parameterName'],
            Value=event['parameterValue'],
            Type='String',
            Overwrite=True
        )

    reponseString = 'Updated parameter %s with value %s.' % (event['parameterName'],
    event['parameterValue'])

    return reponseString
```

8. 在 Lambda function handler and role 部分的 Role 列表中，选择您在任务 2 中为 Lambda 创建的服务角色。
9. 选择 Next，然后选择 Create function。
10. 要测试 Lambda 函数，请从 Actions 菜单选择 Configure Test Event。
11. 使用以下 JSON 替换现有文本。

```
{
  "parameterName": "latestAmi",
  "parameterValue": "your AMI ID"
}
```

12. 选择 Save and test。输出应说明已成功更新参数并包括有关更新的详细信息。例如，“已使用值 ami-123456 更新参数 latestAmi”。

任务 4：创建 Automation 文档并修补 AMI

使用以下过程创建并运行 Automation 文档，该文档修补您为 latestAmi 参数指定的 AMI。在 Automation 工作流完成后，使用新修补的 AMI 的 ID 更新 latestAmi 的值。接下来的执行过程会使用以前执行创建的 AMI。

创建自动化文档并修补 AMI ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择创建文档。
4. 在 Name 字段中，键入 UpdateMyLatestWindowsAmi。
5. 在文档类型列表中，选择自动化文档。
6. 删除 Content 字段中的方括号，然后将以下 JSON 实例文档粘贴到此处。

Note

您必须使用在 [设置 Automation \(p. 107\)](#) 时创建的服务角色 ARN 和实例配置文件角色，来更改此示例中 *assumeRole* 和 *IamInstanceProfileName* 的值。

```
{
  "description": "Systems Manager Automation Demo - Patch AMI and Update SSM Param",
  "schemaVersion": "0.3",
  "assumeRole": "the role ARN you created",
  "parameters": {
    "sourceAMIId": {
      "type": "String",
      "description": "AMI to patch",
      "default": "{{ssm:latestAmi}}"
    },
    "targetAMIname": {
      "type": "String",
      "description": "Name of new AMI",
      "default": "patchedAMI-{{global:DATE_TIME}}"
    }
  },
  "mainSteps": [
    {
      "name": "startInstances",
      "action": "aws:runInstances",
      "timeoutSeconds": 1200,
      "maxAttempts": 1,
      "onFailure": "Abort",
      "inputs": {
        "ImageId": "{{ sourceAMIId }}",
        "InstanceType": "m3.large",
        "MinInstanceCount": 1,
        "MaxInstanceCount": 1,
        "IamInstanceProfileName": "the name of the IAM role you created"
      }
    },
    {
      "name": "installMissingWindowsUpdates",
      "action": "aws:runCommand",
      "maxAttempts": 1,
      "onFailure": "Continue",
      "inputs": {
        "DocumentName": "AWS-InstallMissingWindowsUpdates",
```

```

        "InstanceIds":[
            "{{ startInstances.InstanceIds }}"
        ],
        "Parameters":{
            "UpdateLevel":"Important"
        }
    },
    {
        "name":"stopInstance",
        "action":"aws:changeInstanceState",
        "maxAttempts":1,
        "onFailure":"Continue",
        "inputs":{
            "InstanceIds":[
                "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"stopped"
        }
    },
    {
        "name":"createImage",
        "action":"aws:createImage",
        "maxAttempts":1,
        "onFailure":"Continue",
        "inputs":{
            "InstanceId":"{{ startInstances.InstanceIds }}",
            "ImageName":"{{ targetAMIname }}",
            "NoReboot":true,
            "ImageDescription":"AMI created by EC2 Automation"
        }
    },
    {
        "name":"terminateInstance",
        "action":"aws:changeInstanceState",
        "maxAttempts":1,
        "onFailure":"Continue",
        "inputs":{
            "InstanceIds":[
                "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState":"terminated"
        }
    },
    {
        "name":"updateSsmParam",
        "action":"aws:invokeLambdaFunction",
        "timeoutSeconds":1200,
        "maxAttempts":1,
        "onFailure":"Abort",
        "inputs":{
            "FunctionName":"Automation-UpdateSsmParam",
            "Payload":"{\"parameterName\":\"latestAmi\", \"parameterValue\":\
                \"{{createImage.ImageId}}\"}"
        }
    },
    "outputs":[
        "createImage.ImageId"
    ]
}

```

7. 选择创建文档以保存文档。
8. 在导航窗格中，选择自动化，然后选择执行自动化。
9. 在自动化文档列表中，选择 UpdateMyLatestWindowsAmi。

10. 在文档详细信息部分，确认文档版本是否设置为 1。
11. 在执行模式部分中，选择一次执行整个自动化。
12. 保留目标和速率控制选项为禁用状态。
13. 执行完成后，在导航窗格中选择 Parameter Store，并确认 latestAmi 的新值与 Automation 工作流返回的值匹配。您还可以验证新 AMI ID 与 EC2 控制台的 AMI 部分中的 Automation 输出匹配。

将 Automation 用于 Jenkins

如果您的组织在 CI/CD 管道中使用 Jenkins 软件，则您可以添加 Automation 作为构建后步骤，用于将应用程序发行版预安装到 Amazon Machine Image (AMI) 中。您还可以使用 Jenkins 计划功能来调用 Automation 并创建自己的操作系统 (OS) 修补计划。

以下示例显示如何从运行在本地或 Amazon EC2 上的 Jenkins 服务器调用 Automation。对于身份验证，Jenkins 服务器使用 AWS 凭证，该凭证基于您在示例中创建的 AWS Identity and Access Management (IAM) 用户。如果您的 Jenkins 服务器运行在 Amazon EC2 中，则还可以使用 IAM 实例配置文件角色对其进行身份验证。

Note

配置您的实例时，请确保遵循 Jenkins 安全最佳实践。

开始前的准备工作

在您配置 Automation 用于 Jenkins 之前，请完成以下任务。

- 完成 [使用 Automation、Lambda 和 Parameter Store 简化 AMI 修补 \(p. 141\)](#) 示例。以下示例使用在该示例中创建的 UpdateMyLatestWindowsAmi Automation 文档。
- 为 Automation 配置 IAM 角色。Systems Manager 需要实例配置文件角色和服务角色 ARN 来处理 Automation 工作流。有关更多信息，请参阅 [设置 Automation \(p. 107\)](#)。
- 在您为 Automation 配置 IAM 角色之后，使用以下过程为 Jenkins 服务器创建 IAM 用户账户。在执行期间，Automation 工作流使用 IAM 用户账户的访问密钥和私有密钥对 Jenkins 服务器进行身份验证。

为 Jenkins 服务器创建用户账户

1. 从 [IAM 控制台](#) 上的 Users 页面中，选择 Add User。
2. 在 Set user details 部分中指定用户名 (例如，Jenkins)。
3. 在 Select AWS access type 部分中，选择 Programmatic Access。
4. 选择下一步：权限。
5. 在 Set permissions for 部分，选择 Attach existing policies directly。
6. 在“Filter”字段中，键入 AmazonSSMFullAccess。
7. 选中该策略旁边的复选框，然后选择 Next:Review。
8. 验证详细信息，然后选择 Create。
9. 将访问密钥和私有密钥复制到文本文件。您可以在接下来的过程中指定这些凭证。

使用以下过程在 Jenkins 服务器上配置 AWS CLI。

为 Automation 配置 Jenkins 服务器

1. 如果尚未安装，请下载 AWS CLI 到您的 Jenkins 服务器。有关更多信息，请参阅 [安装 AWS Command Line Interface](#)。
2. 在 Jenkins 服务器的终端窗口中，运行以下命令以配置 AWS CLI。

```
sudo su - jenkins
aws configure
```

3. 出现提示时，输入您在 IAM 中创建 Jenkins 用户时收到的 AWS 访问密钥和私有密钥。指定默认区域。有关配置 AWS CLI 的更多信息，请参阅[配置 AWS Command Line Interface](#)。

使用以下过程配置您的 Jenkins 项目来调用 Automation。

配置 Jenkins 服务器以调用 Automation

1. 在 Web 浏览器中打开 Jenkins 控制台。
2. 选择要配置用于 Automation 的项目，然后选择 Configure。
3. 在 Build 选项卡上，选择 Add Build Step。
4. 选择 Execute shell 或 Execute Windows batch command (具体取决于您的操作系统)。
5. 在 Command (命令) 框中，运行 AWS CLI 命令，如下所示：

```
aws --region the region of your source AMI ssm start-automation-execution --document-name your document name --parameters parameters for the document
```

以下示例命令使用 UpdateMyLatestWindowsAmi 文档以及在[使用 Automation、Lambda 和 Parameter Store 简化 AMI 修补 \(p. 141\)](#)中创建的 Systems Manager 参数 latestAmi：

```
aws --region region-id ssm start-automation-execution \
  --document-name UpdateMyLatestWindowsAmi \
  --parameters \
    "sourceAMIId='{{ssm:latestAmi}}'"
```

在 Jenkins 中，命令类似于以下屏幕截图中的示例。



6. 在 Jenkins 项目中，选择 Build Now。Jenkins 会返回与以下示例类似的输出。

Console Output

```
Started by user admin
Building in workspace /var/lib/jenkins/workspace/Build AMI
[Build AMI] $ /bin/sh -xe /tmp/hudson3259912997441414819.sh
+ aws --region us-east-1 ssm start-automation-execution --document-name UpdateMyLatestWindowsAmi --parameters 'sourceAMIId='\''{{ssm:latestAmi}}'\''
{
  "AutomationExecutionId": "7badf13a-ff8c-11e6-9503-9d48daa849f3"
}
Finished: SUCCESS
```

修补 AMI 和更新 Auto Scaling 组

以下示例建立在 [使用 Automation、Lambda 和 Parameter Store 简化 AMI 修补 \(p. 141\)](#) 示例的基础上，并增加了使用新修补的 AMI 来更新 Auto Scaling 组的步骤。此过程确保新映像自动可供使用 Auto Scaling 组的不同计算环境使用。

此示例中 Automation 工作流的最后一步使用 AWS Lambda 函数复制现有启动配置并将 AMI ID 设置为新修补的 AMI。然后，使用新启动配置更新 Auto Scaling 组。在此类型的 Auto Scaling 方案中，用户可以终止 Auto Scaling 组中的现有实例以强制启动使用新映像的实例。或者，用户可以等待以允许缩减或扩展事件正常启动较新的实例。

开始前的准备工作

在开始本示例之前，请完成以下任务。

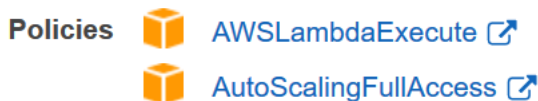
- 完成 [使用 Automation、Lambda 和 Parameter Store 简化 AMI 修补 \(p. 141\)](#) 示例。以下示例使用在该示例中创建的 UpdateMyLatestWindowsAmi Automation 文档。
- 为 Automation 配置 IAM 角色。Systems Manager 需要实例配置文件角色和服务角色 ARN 来处理 Automation 工作流。有关更多信息，请参阅 [设置 Automation \(p. 107\)](#)。
- 如果您不熟悉 Lambda，建议您使用 AWS Lambda Developer Guide 中的 [创建简单的 Lambda 函数](#) 主题创建简单的 Lambda 函数。此主题将帮助您详细了解创建 Lambda 函数所需的一些步骤。

任务 1：为 AWS Lambda 创建 IAM 角色

使用以下过程为 AWS Lambda 创建 IAM 服务角色。此角色包括 AWSLambdaExecute 和 AutoScalingFullAccess 托管策略。这些策略授予 Lambda 权限，以便使用 Lambda 函数通过最新修补的 AMI 创建新 Auto Scaling 组。

为 Lambda 创建 IAM 服务角色

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择 Roles，然后选择 Create Role。
3. 在选择受信任实体的类型页面上的 AWS 服务下，选择 Lambda。
4. 在选择您的使用案例部分中，依次选择 Lambda 和下一步: 权限。
5. 在附加权限策略页面上，搜索 AWSLambdaExecute，然后选择它旁边的选项。搜索 AutoScalingFullAccess，然后选择它旁边的选项。
6. 选择下一步: 审核。
7. 在审核页面上，验证 AWSLambdaExecute 和 AutoScalingFullAccess 是否列在策略下。



8. 在角色名称框中键入名称，然后键入描述。
9. 选择 Create role。系统将让您返回到 Roles 页。

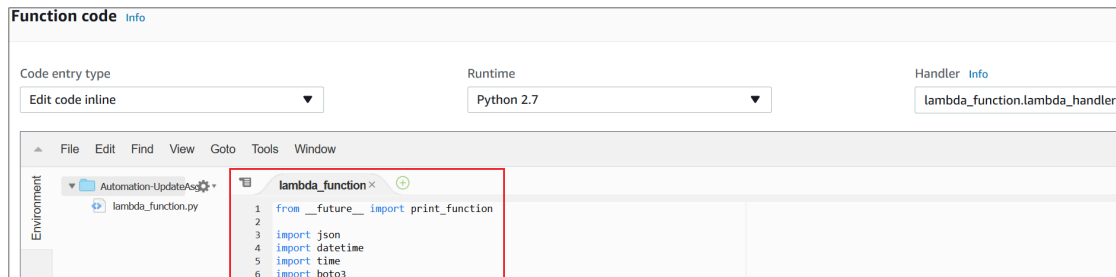
步骤 2：创建 AWS Lambda 函数

使用以下过程创建新 Lambda 函数，该函数字段使用最新修补的 AMI 创建新 Auto Scaling 组。

创建 Lambda 函数

1. Sign in to the AWS Management Console and open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.

2. 选择 Create function。
3. 验证从头开始创作是否处于选中状态。
4. 在名称字段中，键入 Automation-UpdateAsg。
5. 在 Runtime 列表中，选择 Python 2.7。
6. 在角色列表中，验证选择现有角色是否处于选中状态。
7. 在现有角色列表中，选择您先前创建的角色。
8. 选择 Create function。系统将显示 Automation-UpdateSAsg 的代码和配置页面。
9. 在 Designer 部分中不进行任何更改。
10. 在函数代码部分中，删除 lambda_function 字段中预填充的代码，然后粘贴以下代码示例。



```
from __future__ import print_function

import json
import datetime
import time
import boto3

print('Loading function')

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    # get autoscaling client
    client = boto3.client('autoscaling')

    # get object for the ASG we're going to update, filter by name of target ASG
    response =
client.describe_auto_scaling_groups(AutoScalingGroupNames=[event['targetASG']])

    if not response['AutoScalingGroups']:
        return 'No such ASG'

    # get name of InstanceID in current ASG that we'll use to model new Launch
    Configuration after
    sourceInstanceId = response.get('AutoScalingGroups')[0]['Instances'][0]
['InstanceId']

    # create LC using instance from target ASG as a template, only diff is the name of
    the new LC and new AMI
    timeStamp = time.time()
    timeStampString = datetime.datetime.fromtimestamp(timeStamp).strftime('%Y-%m-%d
%H-%M-%S')
    newLaunchConfigName = 'LC ' + event['newAmiID'] + ' ' + timeStampString
    client.create_launch_configuration(
        InstanceId = sourceInstanceId,
        LaunchConfigurationName=newLaunchConfigName,
        ImageId= event['newAmiID'] )

    # update ASG to use new LC
```

```
response = client.update_auto_scaling_group(AutoScalingGroupName =
event['targetASG'],LaunchConfigurationName = newLaunchConfigName)

return 'Updated ASG `%s` with new launch configuration `%s` which includes AMI `
%s`.' % (event['targetASG'], newLaunchConfigName, event['newAmiID'])
```

11. 指定此页面上的其余配置选项。
12. 选择 Save (保存)。
13. 选择 Test。
14. 在配置测试事件页面中，验证创建新测试事件是否处于选中状态。
15. 在事件模板列表中，验证 Hello World 是否处于选中状态。
16. 在事件名称字段中，键入名称。
17. 将现有示例替换为以下 JSON。输入 AMI ID 和 Auto Scaling 组。

```
{
  "newAmiID":"valid AMI ID",
  "targetASG":"name of your Auto Scaling group"
}
```

18. 选择 Save (保存)。
19. 选择 Test。输出表明，已使用新的启动配置成功更新 Auto Scaling 组。

任务 3：创建一个 Automation 文档，修补 AMI，并更新 Auto Scaling 组

使用以下过程创建并运行 Automation 文档，该文档修补您为 latestAmi 参数指定的 AMI。然后，Automation 工作流更新 Auto Scaling 组以使用最新的修补后的 AMI。

创建并运行自动化文档 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择创建文档。
4. 在 Name 字段中，键入 PatchAmiandUpdateAsg。
5. 在文档类型列表中，选择自动化文档。
6. 删除 Content 字段中的方括号，然后将以下 JSON 实例文档粘贴到此处。

Note

您必须使用在 [设置 Automation \(p. 107\)](#) 时创建的服务角色 ARN 和实例配置文件角色，来更改此示例中 *assumeRole* 和 *IamInstanceProfileName* 的值。

```
{
  "description":"Systems Manager Automation Demo - Patch AMI and Update ASG",
  "schemaVersion":"0.3",
  "assumeRole":"the service role ARN you created",
  "parameters":{
    "sourceAMIid":{
      "type":"String",
      "description":"AMI to patch"
    },
  },
}
```

```

"targetAMIname":{
  "type":"String",
  "description":"Name of new AMI",
  "default":"patchedAMI-{{global:DATE_TIME}}"
},
"targetASG":{
  "type":"String",
  "description":"Autoscaling group to Update"
}
},
"mainSteps":[
  {
    "name":"startInstances",
    "action":"aws:runInstances",
    "timeoutSeconds":1200,
    "maxAttempts":1,
    "onFailure":"Abort",
    "inputs":{
      "ImageId":"{{ sourceAMIid }}",
      "InstanceType":"m3.large",
      "MinInstanceCount":1,
      "MaxInstanceCount":1,
      "IamInstanceProfileName":"the name of the instance IAM role you created"
    }
  },
  {
    "name":"installMissingWindowsUpdates",
    "action":"aws:runCommand",
    "maxAttempts":1,
    "onFailure":"Continue",
    "inputs":{
      "DocumentName":"AWS-InstallMissingWindowsUpdates",
      "InstanceIds":[
        "{{ startInstances.InstanceIds }}"
      ],
      "Parameters":{
        "UpdateLevel":"Important"
      }
    }
  },
  {
    "name":"stopInstance",
    "action":"aws:changeInstanceState",
    "maxAttempts":1,
    "onFailure":"Continue",
    "inputs":{
      "InstanceIds":[
        "{{ startInstances.InstanceIds }}"
      ],
      "DesiredState":"stopped"
    }
  },
  {
    "name":"createImage",
    "action":"aws:createImage",
    "maxAttempts":1,
    "onFailure":"Continue",
    "inputs":{
      "InstanceId":"{{ startInstances.InstanceIds }}",
      "ImageName":"{{ targetAMIname }}",
      "NoReboot":true,
      "ImageDescription":"AMI created by EC2 Automation"
    }
  },
  {
    "name":"terminateInstance",

```

```
        "action": "aws:changeInstanceState",
        "maxAttempts": 1,
        "onFailure": "Continue",
        "inputs": {
            "InstanceIds": [
                "{{ startInstances.InstanceIds }}"
            ],
            "DesiredState": "terminated"
        }
    },
    {
        "name": "updateASG",
        "action": "aws:invokeLambdaFunction",
        "timeoutSeconds": 1200,
        "maxAttempts": 1,
        "onFailure": "Abort",
        "inputs": {
            "FunctionName": "Automation-UpdateAsg",
            "Payload": "{\"targetASG\": \"{{targetASG}}\", \"newAmiID\": \"{{createImage.ImageId}}\"}"
        }
    },
    "outputs": [
        "createImage.ImageId"
    ]
}
```

7. 选择创建文档以保存文档。
8. 选择自动化，然后选择执行自动化。
9. 在自动化文档列表中，选择 PatchAmiandUpdateAsg。
10. 在文档详细信息部分，确认文档版本是否设置为 1。
11. 在执行模式部分中，选择一次执行整个自动化。
12. 保留目标和速率控制选项为禁用状态。
13. 为 sourceAMId 指定 Windows AMI ID，为 targetASG 指定您的 Auto Scaling 组名。
14. 选择执行自动化。
15. 执行完成后，在 Amazon EC2 控制台中，选择 Auto Scaling，然后选择 Launch Configurations。验证您看到了新的启动配置，并且使用了新 AMI ID。
16. 选择 Auto Scaling，然后选择 Auto Scaling Groups。验证 Auto Scaling 组使用了新的启动配置。
17. 终止 Auto Scaling 组中的一个或多个实例。使用新 AMI ID 替换将要启动的实例。

Note

您可以通过编辑 Lambda 函数正常终止实例，从而进一步自动完成新 AMI 的部署。您还可以调用自己的 Lambda 函数并利用 AWS CloudFormation 的功能来更新 Auto Scaling 组。有关更多信息，请参阅 [UpdatePolicy Attribute](#)。

Automation 系统变量

Systems Manager Automation 文档使用以下变量。有关如何使用这些变量的示例，请查看 AWS-UpdateWindowsAmi 文档的 JSON 源。

Note

以下过程描述了您在 Amazon EC2 控制中执行的步骤。您也可以在新的 [AWS Systems Manager 控制台](#) 中执行这些步骤。新控制台中的步骤将不同于下面的步骤。

查看 AWS-UpdateWindowsAmi 文档的 JSON 源。

1. 在 Amazon EC2 控制台中，展开 Systems Manager Shared Resources，然后选择 Documents。
2. 选择 AWS-UpdateWindowsAmi。
3. 在下方窗格中，选择 Content 选项卡。

系统变量

Automation 文档目前支持以下系统变量。

变量	详细信息
global:ACCOUNT_ID	在其中运行自动化的 AWS Identity and Access Management (IAM) 用户或角色的 AWS 账户 ID。
global:DATE	格式为 yyyy-MM-dd 的日期 (执行时间)。
global:DATE_TIME	格式为 yyyy-MM-dd_HH.mm.ss 的日期和时间 (执行时间)。
global:REGION	在其中运行文档的区域。例如，us-east-2。

Automation 变量

Automation 文档目前支持以下自动化变量。

变量	详细信息
automation:EXECUTION_ID	分配给当前自动化执行的唯一标识符。例如，1a2b3c-1a2b3c-1a2b3c-1a2b3c1a2b3c1a2b3c。

主题

- [术语 \(p. 153\)](#)
- [支持的场景 \(p. 156\)](#)
- [不支持的场景 \(p. 158\)](#)

术语

以下术语描述了如何解析变量和参数。

术语	定义	示例
常量 ARN	没有变量的有效 ARN	arn:aws:iam::123456789012:role/roleName
文档参数	在 Automation 文档的文档级别定义的参数 (例如，instanceId)。可在替换基本字符串时使用该参数。系统会在启动执行时间提供该参数的值。	<pre>{ "description": "Create Image Demo", "version": "0.3", "assumeRole": "\"Your_Automation_Assume_Role_ARN\"", "parameters":{ "instanceId": {</pre>

术语	定义	示例
		<pre>"type": "STRING", "description": "Instance to create image from" } }</pre>
系统变量	在评估文档的任何部分时，被替换到文档中的常规变量。	<pre>"activities": [{ "id": "copyImage", "activityType": "AWS- CopyImage", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "imageName": "{{imageName}}", "SourceImageId": "{{sourceImageId}}", "SourceRegion": "{{sourceRegion}}", "Encrypted": true, "ImageDescription": "Test CopyImage Description created on {{global:DATE}}" } }]</pre>
Automation 变量	在评估文档的任何部分时，被替换到文档中且与自动化执行相关的变量。	<pre>{ "name": "runFixedCmds", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS- RunPowerShellScript", "InstanceIds": ["{{LaunchInstance.InstanceIds}}"], "Parameters": { "commands": ["dir", "date", "echo {Hello {{ssm:administratorName}}", ""{{outputFormat}}" -f "left","right","{{global:DATE}}","{{auto } } }</pre>

术语	定义	示例
SSM 参数	参数服务中定义的变量。该参数未被声明为文档参数。它可能需要相应访问权限。	<pre>{ "description": "Run Command Demo", "schemaVersion": "0.3", "assumeRole": "arn:aws:iam::123456789012:role/ roleName", "parameters": { "commands": { "type": "STRING_LIST", "description": "list of commands to run as part of first step" }, "instanceIds": { "type": "STRING_LIST", "description": "list of instances to run commands on" } }, "mainSteps": [{ "name": "runFixedCmds", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS-RunPowerShellScript", "InstanceIds": ["{LaunchInstance.InstanceIds}"], "Parameters": { "commands": ["dir", "date", "echo {Hello {{ssm:administratorName}}}", "{{outputFormat}}" -f "left","right","{{global:DATE}}","{{auto] } } }] }</pre>

支持的场景

场景	注释	示例
创建时的常量 ARN assumeRole	将执行授权检查，以确定是否允许调用用户传递给定的代入角色。	<pre>{ "description": "Test all Automation resolvable parameters", "schemaVersion": "0.3", "assumeRole": "arn:aws:iam::123456789012:role/ roleName", "parameters": { ... } }</pre>
创建时为 assumeRole 提供的文档参数	必须在文档的参数列表中定义。	<pre>{ "description": "Test all Automation resolvable parameters", "schemaVersion": "0.3", "assumeRole": "{{dynamicARN}}", "parameters": { ... } }</pre>
启动时为文档参数提供的值。	客户提供要用于参数的值。需在文档的参数列表中定义在启动时提供的所有执行输入。	<pre>... "parameters": { "amiId": { "type": "STRING", "default": "ami-7f2e6015", "description": "list of commands to run as part of first step" }, ... }</pre> <p>启动 Automation 执行的输入包括：{"amiId": ["ami-12345678"]}</p>
在步骤定义中引用的 SSM 参数	客户账户中存在的变量，且文档的 assumeRole 有权访问该变量。创建时会执行检查，以确认 assumeRole 是否拥有访问权限。无需在文档的参数列表中设置 SSM 参数。	<pre>... "mainSteps": [{ "name": "RunSomeCommands", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS:RunPowerShell", "InstanceIds": [{"LaunchInstance.InstanceIds"}], "Parameters": { ... } } }] "commands" : [...]</pre>

场景	注释	示例
		<pre>"echo {Hello {{ssm:administratorName}}}"] } } }, ... </pre>
在步骤定义中引用的系统变量	<p>执行时被替换到文档中的系统变量。注入到文档中的值与替换发生的时间相关。举例来说，由于在运行步骤之间需要花费一定时间，因此在步骤 1 中注入的时间变量的值将与在步骤 3 中注入的值不同。无需在文档的参数列表中设置系统变量。</p>	<pre>... "mainSteps": [{ "name": "RunSomeCommands", "action": "aws:runCommand", "aws:runCommand": { "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS:RunPowerShell", "InstanceIds": ["{{LaunchInstance.InstanceIds}}"], "Parameters": { "commands" : ["echo {The time is now {{global:TIME}}}"] } } }, ... </pre>
在步骤定义中引用的 Automation 变量。	<p>无需在文档的参数列表中设置 Automation 变量。唯一的受支持 Automation 变量是 automation:EXECUTION_ID。</p>	<pre>... "mainSteps": [{ "name": "invokeLambdaFunction", "action": "aws:invokeLambdaFunction", "aws:invokeLambdaFunction": { "maxAttempts": 1, "onFailure": "Continue", "inputs": { "FunctionName": "Hello-World- LambdaFunction", "Payload" : "{ "executionId" : "{{automation:EXECUTION_ID}}"}" } } }, ... </pre>

场景	注释	示例
在下一步的定义中参考上一步的输出。	这是一个参数重定向。可使用语法 <code>{{stepName.OutputName}}</code> 引用上一步的输出。客户不能将该语法用于文档参数。系统会在执行引用步骤时解决这个问题。文档参数中未列出该参数。	<pre> ... "mainSteps": [{ "name": "LaunchInstance", "action": "aws:runInstances", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "ImageId": "{{amiId}}", "MinInstanceCount": 1, "MaxInstanceCount": 2 } }, { "name": "changeState", "action": "aws:changeInstanceState", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "InstanceIds": ["{{LaunchInstance.InstanceIds}}"], "DesiredState": "terminated" } } } ... </pre>

不支持的场景

场景	评论	示例
创建时为 <code>assumeRole</code> 提供的 SSM 参数	不支持。	<pre> ... { "description": "Test all Automation resolvable parameters", "schemaVersion": "0.3", "assumeRole": "{{ssm:administratorRoleARN}}", "parameters": { ... } } </pre>
启动时为文档参数提供的 SSM 参数	用户在启动时提供了输入参数，而该参数是一个 SSM 参数	<pre> ... "parameters": { "amiId": { </pre>

场景	评论	示例
		<pre> "type": "STRING", "default": "ami-7f2e6015", "description": "list of commands to run as part of first step" }, ... User supplies input : { "amiId" : "{{ssm:goldenAMIId}}" }</pre>
变量步骤定义	文档中步骤的定义由变量构建而成。	<pre> ... "mainSteps": [{ "name": "LaunchInstance", "action": "aws:runInstances", "{{attemptModel}}": 1, "onFailure": "Continue", "inputs": { "ImageId": "ami-12345678", "MinInstanceCount": 1, "MaxInstanceCount": 2 } } ... User supplies input : { "attemptModel" : "minAttempts" }</pre>
交叉引用文档参数	用户在启动时提供了输入参数，而该参数引用了文档中的另一参数。	<pre> ... "parameters": { "amiId": { "type": "STRING", "default": "ami-7f2e6015", "description": "list of commands to run as part of first step" }, "otherAmiId": { "type": "STRING", "description": "The other amiId to try if this one fails". "default" : "{{amiId}}" }, ...</pre>

场景	评论	示例
多层扩展	文档定义了评估变量名称的变量。它位于变量分隔符 (即 {{ }}) 内，并扩展为变量/参数的值。	<pre>... "parameters": { "param1": { "type": "STRING", "default": "param2", "description": "The parameter to reference" }, "param2": { "type": "STRING", "default" : "echo {Hello world}", "description": "What to run" } }, "mainSteps": [{ "name": "runFixedCmds", "action": "aws:runCommand", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "DocumentName": "AWS-RunPowerShellScript", "InstanceIds" : "{{LaunchInstance.InstanceIds}}", "Parameters": { "commands": ["{{ {{param1}} }}"] } } } ... Note: The customer intention here would be to run a runCommand of "echo {Hello world}"</pre>

排除 Systems Manager Automation 的故障

利用以下信息帮助您排除 Automation 服务中的问题。本主题介绍了依据 Automation 错误消息解决问题的特定任务。

主题

- [常见 Automation 错误 \(p. 161\)](#)
- [Automation 执行无法启动 \(p. 168\)](#)
- [执行已启动，但是状态为“失败” \(p. 169\)](#)
- [执行已启动，但超时 \(p. 170\)](#)

常见 Automation 错误

此部分提供有关常见 Automation 错误的信息。

VPC not defined 400

默认情况下，当 Automation 运行 AWS-UpdateLinuxAmi 文档或 AWS-UpdateWindowsAmi 文档时，系统会在默认 VPC (172.30.0.0/16) 中创建一个临时实例。如果您删除了默认 VPC，会收到以下错误：

VPC not defined 400

要解决此问题，您必须创建一个新的包含子网 ID 的 Automation 文档。复制下面的包含子网 ID 参数的示例文档，并创建一个新文档。有关创建文档的信息，请参阅[演练：创建一个 Automation 文档 \(p. 128\)](#)。

AWS-UpdateLinuxAmi

```
{
  "schemaVersion": "0.3",
  "description": "Updates AMI with Linux distribution packages and Amazon software. For details, see https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/sysman-ami-walkthrough.html",
  "assumeRole": "{{AutomationAssumeRole}}",
  "parameters": {
    "SourceAmiId": {
      "type": "String",
      "description": "(Required) The source Amazon Machine Image ID."
    },
    "InstanceIamRole": {
      "type": "String",
      "description": "(Required) The name of the role that enables Systems Manager (SSM) to manage the instance.",
      "default": "ManagedInstanceProfile"
    },
    "AutomationAssumeRole": {
      "type": "String",
      "description": "(Required) The ARN of the role that allows Automation to perform the actions on your behalf.",
      "default": "arn:aws:iam::{{global:ACCOUNT_ID}}:role/AutomationServiceRole"
    },
    "SubnetId": {
      "type": "String",
      "description": "(Required) The subnet that the created instance will be placed into."
    },
    "TargetAmiName": {
      "type": "String",
      "description": "(Optional) The name of the new AMI that will be created. Default is a system-generated string including the source AMI id, and the creation time and date.",
      "default": "UpdateLinuxAmi_from_{{SourceAmiId}}_on_{{global:DATE_TIME}}"
    },
    "InstanceType": {
      "type": "String",
      "description": "(Optional) Type of instance to launch as the workspace host. Instance types vary by region. Default is t2.micro.",
      "default": "t2.micro"
    },
    "PreUpdateScript": {
      "type": "String",
      "description": "(Optional) URL of a script to run before updates are applied. Default (\"none\") is to not run a script.",
      "default": "none"
    },
    "PostUpdateScript": {
```

```

        "type": "String",
        "description": "(Optional) URL of a script to run after package updates are
        applied. Default (\"none\") is to not run a script.",
        "default": "none"
    },
    "IncludePackages": {
        "type": "String",
        "description": "(Optional) Only update these named packages. By default (\"all\"),
        all available updates are applied.",
        "default": "all"
    },
    "ExcludePackages": {
        "type": "String",
        "description": "(Optional) Names of packages to hold back from updates, under all
        conditions. By default (\"none\"), no package is excluded.",
        "default": "none"
    }
},
"mainSteps": [
    {
        "name": "launchInstance",
        "action": "aws:runInstances",
        "maxAttempts": 3,
        "timeoutSeconds": 1200,
        "onFailure": "Abort",
        "inputs": {
            "ImageId": "${SourceAmiId}",
            "InstanceType": "${InstanceType}",
            "SubnetId": "${ SubnetId }",

            "UserData": "IyEvYmluL2Jhc2gNCg0KZnVuY3Rpb24gZ2V0X2NvbnRlbnRzKCKgew0KICAgIGlmIFsgLXggIiQod2hpY2ggY3Vybm90IC1kUONSSVBUX05BTUUidQogIEZJTTEVfU0laRT0kKGR1IC1rIC90bXAvJFNDUklQVF90QU1FIHwgY3V0IC1mMSkNCiAgZWNobyBBV1
            "MinInstanceCount": 1,
            "MaxInstanceCount": 1,
            "IamInstanceProfileName": "${InstanceIamRole}"
        }
    },
    {
        "name": "updateOSSoftware",
        "action": "aws:runCommand",
        "maxAttempts": 3,
        "timeoutSeconds": 3600,
        "onFailure": "Abort",
        "inputs": {
            "DocumentName": "AWS-RunShellScript",
            "InstanceIds": [
                "${launchInstance.InstanceIds}"
            ],
            "Parameters": {
                "commands": [
                    "set -e",
                    "[ -x \"$(which wget)\" ] && get_contents='wget $1 -O -'",
                    "[ -x \"$(which curl)\" ] && get_contents='curl -s -f $1'",
                    "eval $get_contents https://aws-ssm-downloads-
                    ${global:REGION}.s3.amazonaws.com/scripts/aws-update-linux-instance > /tmp/aws-update-
                    linux-instance",
                    "chmod +x /tmp/aws-update-linux-instance",
                    "/tmp/aws-update-linux-instance --pre-update-script '{{PreUpdateScript}}'
                    --post-update-script '{{PostUpdateScript}}' --include-packages '{{IncludePackages}}' --
                    exclude-packages '{{ExcludePackages}}' 2>&1 | tee /tmp/aws-update-linux-instance.log"
                ]
            }
        }
    },
    {
        "name": "stopInstance",
    }
]

```

```
        "action": "aws:changeInstanceState",
        "maxAttempts": 3,
        "timeoutSeconds": 1200,
        "onFailure": "Abort",
        "inputs": {
            "InstanceIds": [
                "{{launchInstance.InstanceIds}}"
            ],
            "DesiredState": "stopped"
        }
    },
    {
        "name": "createImage",
        "action": "aws:createImage",
        "maxAttempts": 3,
        "onFailure": "Abort",
        "inputs": {
            "InstanceId": "{{launchInstance.InstanceIds}}",
            "ImageName": "{{TargetAmiName}}",
            "NoReboot": true,
            "ImageDescription": "AMI Generated by EC2 Automation on {{global:DATE_TIME}}"
        },
        "from": "{{SourceAmiId}}"
    },
    {
        "name": "terminateInstance",
        "action": "aws:changeInstanceState",
        "maxAttempts": 3,
        "onFailure": "Continue",
        "inputs": {
            "InstanceIds": [
                "{{launchInstance.InstanceIds}}"
            ],
            "DesiredState": "terminated"
        }
    }
],
"outputs": [
    "createImage.ImageId"
]
}
```

AWS-UpdateWindowsAmi

```
{
    "schemaVersion": "0.3",
    "description": "Updates a Microsoft Windows AMI. By default it will install all Windows updates, Amazon software, and Amazon drivers. It will then sysprep and create a new AMI. Supports Windows Server 2008 R2 and greater.",
    "assumeRole": "{{ AutomationAssumeRole }}",
    "parameters": {
        "SourceAmiId": {
            "type": "String",
            "description": "(Required) The source Amazon Machine Image ID."
        },
        "IamInstanceProfileName": {
            "type": "String",
            "description": "(Required) The name of the role that enables Systems Manager to manage the instance.",
            "default": "ManagedInstanceProfile"
        },
        "AutomationAssumeRole": {
            "type": "String",
            "description": "(Required) The ARN of the role that allows Automation to perform the actions on your behalf."
        }
    }
}
```

```

        "default": "arn:aws:iam::{{global:ACCOUNT_ID}}:role/AutomationServiceRole"
    },
    "SubnetId": {
        "type": "String",
        "description": "(Required) The subnet that the created instance will be placed
into."
    },
    "TargetAmiName": {
        "type": "String",
        "description": "(Optional) The name of the new AMI that will be created. Default is
a system-generated string including the source AMI id, and the creation time and date.",
        "default": "UpdateWindowsAml_from_{{SourceAmlId}}_on_{{global:DATE_TIME}}"
    },
    "InstanceType": {
        "type": "String",
        "description": "(Optional) Type of instance to launch as the workspace host.
Instance types vary by region. Default is t2.medium.",
        "default": "t2.medium"
    },
    "IncludeKbs": {
        "type": "String",
        "description": "(Optional) Specify one or more Microsoft Knowledge Base (KB)
article IDs to include. You can install multiple IDs using comma-separated values. When
specified, the categories and security level values are ignored. Valid formats: KB9876543
or 9876543.",
        "default": ""
    },
    "ExcludeKbs": {
        "type": "String",
        "description": "(Optional) Specify one or more Microsoft Knowledge Base (KB)
article IDs to exclude. You can exclude multiple IDs using comma-separated values. When
specified, all these KBs are excluded from install process. Valid formats: KB9876543 or
9876543.",
        "default": ""
    },
    "Categories": {
        "type": "String",
        "description": "(Optional) Specify one or more update categories. You can filter
categories using comma-separated values. By default patches for all categories are
selected. If value supplied, the update list is filtered by those values. Options:
Critical Update, Security Update, Definition Update, Update Rollup, Service Pack, Tool,
Update or Driver. Valid formats include a single entry, for example: Critical Update. Or,
you can specify a comma separated list: Critical Update,Security Update,Definition Update.
NOTE: There cannot be any spaces around the commas.",
        "default": ""
    },
    "SeverityLevels": {
        "type": "String",
        "description": "(Optional) Specify one or more MSRC severity levels associated with
an update. You can filter severity levels using comma-separated values. By default patches
for all security levels are selected. If value supplied, the update list is filtered by
those values. Options: Critical, Important, Low, Moderate or Unspecified. Valid formats
include a single entry, for example: Critical. Or, you can specify a comma separated list:
Critical,Important,Low.",
        "default": ""
    },
    "PreUpdateScript": {
        "type": "String",
        "description": "(Optional) A script provided as a string. It will run prior to
installing OS updates.",
        "default": ""
    },
    "PostUpdateScript": {
        "type": "String",
        "description": "(Optional) A script provided as a string. It will run after
installing OS updates.",
    }
}

```

```
        "default":""
    }
},
"mainSteps":[
    {
        "name":"LaunchInstance",
        "action":"aws:runInstances",
        "timeoutSeconds":1800,
        "maxAttempts":3,
        "onFailure":"Abort",
        "inputs":{
            "ImageId":"{{ SourceAmiId }}",
            "InstanceType":"{{ InstanceType }}",
            "SubnetId":"{{ SubnetId }}",
            "MinInstanceCount":1,
            "MaxInstanceCount":1,
            "IamInstanceProfileName":"{{ IamInstanceProfileName }}"
        }
    },
    {
        "name":"RunPreUpdateScript",
        "action":"aws:runCommand",
        "maxAttempts":3,
        "onFailure":"Abort",
        "timeoutSeconds":1800,
        "inputs":{
            "DocumentName":"AWS-RunPowerShellScript",
            "InstanceIds":[
                "{{ LaunchInstance.InstanceIds }}"
            ],
            "Parameters":{
                "commands":"{{ PreUpdateScript }}"
            }
        }
    },
    {
        "name":"UpdateSSMAgent",
        "action":"aws:runCommand",
        "maxAttempts":3,
        "onFailure":"Abort",
        "timeoutSeconds":600,
        "inputs":{
            "DocumentName":"AWS-UpdateSSMAgent",
            "InstanceIds":[
                "{{ LaunchInstance.InstanceIds }}"
            ]
        }
    },
    {
        "name":"UpdateEC2Config",
        "action":"aws:runCommand",
        "maxAttempts":3,
        "onFailure":"Abort",
        "timeoutSeconds":7200,
        "inputs":{
            "DocumentName":"AWS-InstallPowerShellModule",
            "InstanceIds":[
                "{{ LaunchInstance.InstanceIds }}"
            ],
            "Parameters":{
                "executionTimeout":"7200",
                "source":"https://aws-ssm-downloads-{{global:REGION}}.s3.amazonaws.com/
PSModules/AWSUpdateWindowsInstance/Latest/AWSUpdateWindowsInstance.zip",
                "sourceHash":"14CAD416F4A054894EBD2091EA4B99E542368BE5895BDD466B567C1ABA87C87C",
                "commands":[]
            }
        }
    }
]
```

```

        "Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force",
        "Import-Module AWSUpdateWindowsInstance",
        "if ([Environment]::OSVersion.Version.Major -ge 10) {" ,
        "  Install-AwsUwiEC2Launch -Id {{ automation:EXECUTION_ID }}",
        "} else {" ,
        "  Install-AwsUwiEC2Config -Id {{ automation:EXECUTION_ID }}",
        "}"
      ]
    }
  },
  {
    "name": "UpdateAWSPVDriver",
    "action": "aws:runCommand",
    "maxAttempts": 3,
    "onFailure": "Abort",
    "timeoutSeconds": 600,
    "inputs": {
      "DocumentName": "AWS-ConfigureAWSPackage",
      "InstanceIds": [
        "{{ LaunchInstance.InstanceIds }}"
      ],
      "Parameters": {
        "name": "AWSPVDriver",
        "action": "Install"
      }
    }
  },
  {
    "name": "InstallWindowsUpdates",
    "action": "aws:runCommand",
    "maxAttempts": 3,
    "onFailure": "Abort",
    "timeoutSeconds": 14400,
    "inputs": {
      "DocumentName": "AWS-InstallWindowsUpdates",
      "InstanceIds": [
        "{{ LaunchInstance.InstanceIds }}"
      ],
      "Parameters": {
        "Action": "Install",
        "IncludeKbs": "{{ IncludeKbs }}",
        "ExcludeKbs": "{{ ExcludeKbs }}",
        "Categories": "{{ Categories }}",
        "SeverityLevels": "{{ SeverityLevels }}"
      }
    }
  },
  {
    "name": "RunPostUpdateScript",
    "action": "aws:runCommand",
    "maxAttempts": 3,
    "onFailure": "Abort",
    "timeoutSeconds": 1800,
    "inputs": {
      "DocumentName": "AWS-RunPowerShellScript",
      "InstanceIds": [
        "{{ LaunchInstance.InstanceIds }}"
      ],
      "Parameters": {
        "commands": "{{ PostUpdateScript }}"
      }
    }
  },
  {
    "name": "RunSysprepGeneralize",

```

```
"action": "aws:runCommand",
"maxAttempts": 3,
"onFailure": "Abort",
"timeoutSeconds": 7200,
"inputs": {
  "DocumentName": "AWS-InstallPowerShellModule",
  "InstanceIds": [
    "{{ LaunchInstance.InstanceIds }}"
  ],
  "Parameters": {
    "executionTimeout": "7200",
    "source": "https://aws-ssm-downloads-{{global:REGION}}.s3.amazonaws.com/
PSModules/AWSUpdateWindowsInstance/Latest/AWSUpdateWindowsInstance.zip",
    "sourceHash": "14CAD416F4A054894EBD2091EA4B99E542368BE5895BDD466B567C1ABA87C87C",
    "commands": [
      "Set-ExecutionPolicy -ExecutionPolicy Unrestricted -Force",
      "Import-Module AWSUpdateWindowsInstance",
      "Start-AwsUwiSysprep -Id {{ automation:EXECUTION_ID }}"
    ]
  }
},
{
  "name": "StopInstance",
  "action": "aws:changeInstanceState",
  "maxAttempts": 3,
  "timeoutSeconds": 7200,
  "onFailure": "Abort",
  "inputs": {
    "InstanceIds": [
      "{{ LaunchInstance.InstanceIds }}"
    ],
    "CheckStateOnly": false,
    "DesiredState": "stopped"
  }
},
{
  "name": "CreateImage",
  "action": "aws:createImage",
  "maxAttempts": 3,
  "onFailure": "Abort",
  "inputs": {
    "InstanceId": "{{ LaunchInstance.InstanceIds }}",
    "ImageName": "{{ TargetAmiName }}",
    "NoReboot": true,
    "ImageDescription": "Test CreateImage Description"
  }
},
{
  "name": "CreateTags",
  "action": "aws:createTags",
  "maxAttempts": 3,
  "onFailure": "Abort",
  "inputs": {
    "ResourceType": "EC2",
    "ResourceIds": [
      "{{ CreateImage.ImageId }}"
    ],
    "Tags": [
      {
        "Key": "Original_AMI_ID",
        "Value": "Created from {{ SourceAmiId }}"
      }
    ]
  }
}
```



```
    },
    {
      "name": "TerminateInstance",
      "action": "aws:changeInstanceState",
      "maxAttempts": 3,
      "onFailure": "Abort",
      "inputs": {
        "InstanceIds": [
          "{{ LaunchInstance.InstanceIds }}"
        ],
        "DesiredState": "terminated"
      }
    }
  ],
  "outputs": [
    "CreateImage.ImageId"
  ]
}
```

Automation 执行无法启动

如果您没有为 Automation 正确配置 IAM 用户、角色和策略，Automation 执行可能因拒绝访问错误或代入角色无效错误而失败。

拒绝访问

以下示例描述了 Automation 执行因拒绝访问错误无法启动的情况。

对 Systems Manager API 的访问被拒绝

错误消息: User: user arn is not authorized to perform: ssm:StartAutomationExecution on resource: document arn (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: AccessDeniedException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx)

- 可能的原因 1：尝试启动 Automation 执行的 IAM 用户没有调用 StartAutomationExecution API 的权限。要解决此问题，请将所需的 IAM 策略附加到用于启动执行的用户账户。有关更多信息，请参阅 [任务 4：配置用户对 Automation 的访问权限 \(p. 112\)](#)。
- 可能的原因 2：尝试启动 Automation 执行的 IAM 用户具有调用 StartAutomationExecution API 的权限，但是无权使用指定 Automation 文档来调用该 API。要解决此问题，请将所需的 IAM 策略附加到用于启动执行的用户账户。有关更多信息，请参阅 [任务 4：配置用户对 Automation 的访问权限 \(p. 112\)](#)。

由于不具备 PassRole 权限访问被拒

错误消息: User: user arn is not authorized to perform: iam:PassRole on resource: automation assume role arn (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: AccessDeniedException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx)

尝试启动 Automation 执行的 IAM 用户不具有代入角色所需的 PassRole 权限。要解决此问题，请将 iam:PassRole 策略附加到尝试启动 Automation 执行的 IAM 用户的角色。有关更多信息，请参阅 [任务 3：将 iam:PassRole 策略附加到您的 Automation 角色 \(p. 112\)](#)。

代入角色无效

运行自动化时，要么在文档中提供代入角色，要么将代入角色作为文档的一个参数值传递。如果未指定或未正确配置代入角色，可出现不同类型的错误。

代入角色格式错误

错误消息: The format of the supplied assume role ARN is invalid. 代入角色格式不正确。要解决该问题, 请验证 Automation 文档中是否指定了有效的代入角色, 或者在执行 Automation 时指定为运行时参数。

无法代入代入角色

错误消息: The defined assume role is unable to be assumed. (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: InvalidAutomationExecutionParametersException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx)

- 可能的原因 1: 代入角色不存在。要解决该问题, 请创建角色。有关更多信息, 请参阅 [the section called “设置 Automation” \(p. 107\)](#)。创建此角色的特定详细信息在以下主题中介绍 [任务 1: 为 Automation 创建服务角色 \(p. 110\)](#)。
- 可能的原因 2: 代入角色没有与 Systems Manager 服务的信任关系。要解决该问题, 请创建信任关系。有关更多信息, 请参阅 [任务 2: 为 Automation 添加信任关系 \(p. 111\)](#)。

执行已启动, 但是状态为“失败”

特定操作失败

自动化文档包含步骤, 并且步骤按顺序运行。每个步骤调用一项或多项 AWS 服务 API。API 可确定本步的输入、行为和输出。在多个位置错误可能导致步骤失败。失败消息指示出现错误的时间和位置。

要查看 EC2 控制台中的失败消息, 请选择失败步骤的 View Outputs 链接。要查看 CLI 中的失败消息, 请调用 get-automation-execution 并查找失败的 StepExecution 中的 FailureMessage 属性。

在以下示例中, 与 aws:runInstance 操作相关联的步骤失败。每个示例探讨了不同类型的错误。

缺少映像

错误消息: Automation Step Execution fails when it is launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [The image id '[ami id]' does not exist (Service: AmazonEC2; Status Code: 400; Error Code: InvalidAMIID.NotFound; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

aws:runInstances 操作收到的 ImageId 输入不存在。要解决该问题, 请用正确的 AMI ID 更新 automation 文档或参数值。

代入角色策略没有足够的权限

错误消息: Automation Step Execution fails when it is launching the instance(s). Get Exception from RunInstances API of ec2 Service. Exception Message from RunInstances API: [You are not authorized to perform this operation. Encoded authorization failure message: xxxxxxxx (Service: AmazonEC2; Status Code: 403; Error Code: UnauthorizedOperation; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

代入角色的权限不足, 无法在 Amazon EC2 实例上调用 RunInstances API。要解决此问题, 请将 IAM 策略附加到有权调用 RunInstances API 的代入角色。有关更多信息, 请参见 [方法 2: 使用 IAM 为 Automation 配置角色 \(p. 110\)](#)。

意外状态

错误消息: Step fails when it is verifying launched instance(s) are ready to be used. Instance i-xxxxxxx entered unexpected state: shutting-down. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

- 可能的原因 1: 实例或 Amazon EC2 服务有问题。要解决此问题, 请登录到实例或查阅实例系统日志以了解实例启动关闭的原因。
- 可能的原因 2: 为 `aws:runInstances` 操作指定的用户数据脚本有问题或语法错误。验证用户数据脚本的语法。另外, 请验证用户数据脚本是否未关闭实例, 或者调用了关闭该实例的其他脚本。

操作特定失败参考

如果步骤失败, 失败消息可能指示失败时正在调用哪个服务。下表列出每个操作调用的服务。该表还提供指向有关每个服务的信息的链接。

Action	此操作调用的 AWS 服务	有关此服务的信息	内容故障排除
<code>aws:runInstances</code>	Amazon EC2	Amazon EC2 用户指南	EC2 实例故障排除
<code>aws:changeInstanceState</code>	Amazon EC2	Amazon EC2 用户指南	EC2 实例故障排除
<code>aws:runCommand</code>	Systems Manager	Systems Manager Run Command	对 Run Command 进行故障排除
<code>aws:createImage</code>	Amazon EC2	Amazon Machine Images	
<code>aws:createStack</code>	AWS CloudFormation	AWS CloudFormation User Guide	AWS CloudFormation 疑难解答
<code>aws:deleteStack</code>	AWS CloudFormation	AWS CloudFormation User Guide	AWS CloudFormation 疑难解答
<code>aws:deleteImage</code>	Amazon EC2	Amazon Machine Images	
<code>aws:copyImage</code>	Amazon EC2	Amazon Machine Images	
<code>aws:createTag</code>	Amazon EC2, Systems Manager	EC2 资源和标签	
<code>aws:invokeLambdaFunction</code>	AWS Lambda	AWS Lambda Developer Guide	Lambda 故障排除

Automation 服务内部错误

错误消息: Internal Server Error. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

Automation 服务中的一个问题导致指定 Automation 文档无法正确执行。要解决此问题, 请联系 AWS Support。请提供执行 ID 和客户 ID (如果有)。

执行已启动, 但超时

错误消息: Step timed out while step is verifying launched instance(s) are ready to be used. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

`aws:runInstances` 操作中的步骤超时。如果运行本步骤操作所花的时间超出此步骤中为 `timeoutSeconds` 指定的值，就会发生这种情况。要解决该问题，请为 `timeoutSeconds` 指定较长的值。如果不能解决问题，请调查步骤运行时间超出预期的原因。

AWS Systems Manager Run Command

借助 AWS Systems Manager Run Command，您能够以远程方式安全地管理托管实例的配置。托管实例是您混合环境中已经针对 Systems Manager 配置的任意 Amazon EC2 实例或本地计算机。利用 Run Command，您可以自动完成常用管理任务以及大规模执行临时配置更改。您可以从 AWS 控制台、AWS Command Line Interface、AWS Tools for Windows PowerShell 或 AWS 开发工具包使用 Run Command。Run Command 不另外收取费。

管理员使用 Run Command 可以在其托管实例上执行以下类型的任务：安装或引导应用程序，构建部署管道，从 Auto Scaling 组终止实例时捕获日志文件，以及将实例加入 Windows 域等等。

入门

下表包含可帮助您开始使用 Run Command 的信息。

主题	详细信息
教程：远程管理您的 Amazon EC2 实例 (Amazon EC2 用户指南)	(可选) 此教程介绍如何在 AWS Tools for Windows PowerShell 或 AWS Command Line Interface (AWS CLI) 中使用 Run Command 快速发送命令。
Systems Manager 先决条件 (p. 6)	(必需) 验证您的实例是否符合 Run Command 的最低要求，配置所需的角色，以及安装 SSM 代理。
在混合环境中设置 AWS Systems Manager (p. 31)	(可选) 将内部服务器和虚拟机注册到 AWS，以便使用 Run Command 管理。
使用 Systems Manager Run Command 运行命令 (p. 181)	了解如何从 EC2 控制台运行命令，以及如何对托管实例队列运行命令。
Run Command 演练 (p. 191)	了解如何使用 AWS Tools for Windows PowerShell 或 AWS CLI 运行命令。

相关内容

- [在 EC2 实例上远程运行命令](#) (10 分钟教程)
- 有关 Systems Manager 限制的信息，请参阅 [AWS Systems Manager 限制](#)。

内容

- [设置 Run Command](#) (p. 172)
- [使用 Systems Manager Run Command 运行命令](#) (p. 181)
- [了解命令状态](#) (p. 189)
- [Run Command 演练](#) (p. 191)
- [Systems Manager Run Command 疑难解答](#) (p. 201)

相关内容

- [配置对 Systems Manager 的访问权限](#) (p. 9)
- [安装和配置 SSM 代理](#) (p. 14)

- [将 Run Command 配置为 CloudWatch Events 目标 \(p. 174\)](#)
- [Amazon EC2 Systems Manager API Reference](#)

设置 Run Command

您必须为运行命令的任何用户配置 AWS Identity and Access Management (IAM) 用户策略，并为处理命令的任何实例配置 IAM 实例配置文件角色，然后才能使用 Run Command 管理实例。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

本部分还包含用于监控命令执行和限制对标记的实例的命令访问的建议任务。本部分中的任务不是必需的，但它们有助于最大程度地提高您的实例的安全性和减少日常管理。因此，我们强烈建议您完成本部分中的任务。

主题

- [为 Run Command 配置 Amazon CloudWatch Logs \(p. 172\)](#)
- [为 Run Command 配置 CloudWatch Events \(p. 173\)](#)
- [为 Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)
- [基于实例标签限制 Run Command 访问 \(p. 179\)](#)

为 Run Command 配置 Amazon CloudWatch Logs

使用 Run Command 发送命令时，您可以指定要发送命令输出的位置。默认情况下，Systems Manager 仅返回命令输出的前 1200 个字符。如果您要查看命令输出的完整详情，可以指定 Amazon Simple Storage Service (Amazon S3) 存储桶。也可以指定 Amazon CloudWatch Logs。如果您指定 CloudWatch Logs，则 Run Command 会定期向 CloudWatch Logs 发送所有命令输出和错误日志。您可以近乎实时地监控输出日志，搜索特定短语、值或模式，及基于搜索创建警报。

如果您配置了实例或本地混合计算机来使用 AmazonEC2RoleforSSM AWS Identity and Access Management (IAM) 托管策略，则您的实例无需额外配置即可向 CloudWatch Logs 发送输出。如果从控制台发送命令，则您只需选择此选项；或者，如果使用 AWS CLI、Tools for Windows PowerShell 或 API 操作，则需添加 `cloud-watch-output-config` 部分和 `CloudWatchOutputEnabled` 参数。我们将在本主题的稍后内容中详细介绍 `cloud-watch-output-config` 部分和 `CloudWatchOutputEnabled` 参数。

如果您在多个实例上使用自定义策略，则必须在每个实例上更新此类策略，以允许 Systems Manager 向 CloudWatch Logs 发送输出和日志。将以下策略对象添加到您的自定义策略。有关更新 IAM 策略的更多信息，请参阅[编辑 IAM 策略](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream",
    "logs:DescribeLogGroups",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
  ],
  "Resource": "*"
},
```

在您发送命令时指定 CloudWatch Logs

要在您从 AWS 管理控制台发送命令时将 CloudWatch Logs 指定为输出，请在 Output options (输出选项) 部分选择 CloudWatch Output (CloudWatch 输出)。(可选) 您可以在您要发送命令输出的位置指定 CloudWatch Logs 组的名称。如果您未指定组名称，则 Systems Manager 会自动为您创建一个日志组。日志组使用以下命名格式：`aws/ssm/SystemsManagerDocumentName`。

如果您使用 AWS CLI 运行命令，则必须在您的命令中指定 `cloud-watch-output-config` 部分。此部分可让您指定 `CloudWatchOutputEnabled` 参数以及 `CloudWatchLogGroupName` 参数（可选）。示例如下：

```
aws ssm send-command --document-name "AWS-RunPowerShellScript" --parameters commands=["echo  
helloWorld"] --targets "Key=instanceids,Values=an instance ID" --cloud-watch-output-config  
'{"CloudWatchLogGroupName": "log group name", "CloudWatchOutputEnabled": true}'
```

在 CloudWatch Logs 中查看命令输出

一旦命令开始运行，Systems Manager 便会近乎实时地向 CloudWatch Logs 发送输出。CloudWatch Logs 中的输出使用以下格式：

CommandID/InstanceID/PluginID/stdout

CommandID/InstanceID/PluginID/stderr

执行的输出每 30 秒或当缓冲区超过 200 KB 时（以先发生的为准）上传一次。

Note

日志流仅在输出数据可用时创建。例如，如果执行不存在错误数据，则 `stderr` 流不会创建。

下方是在 CloudWatch Logs 中显示的命令输出的示例。

```
Group - /aws/ssm/AWS-RunShellScript  
Streams -  
1234-567-8910/i-abcd-efg-hijk/AWS-RunPowerShellScript/stdout  
24/1234-567-8910/i-abcd-efg-hijk/AWS-RunPowerShellScript/stderr
```

为 Run Command 配置 CloudWatch Events

使用 Amazon CloudWatch Events 记录命令执行状态更改。您可以创建一个规则，只要状态发生变换或者在变换到一个或多个感兴趣的状态时，就运行该规则。

您还可将 Run Command 指定作为发生 CloudWatch 事件时的目标操作。例如，假设触发的 CloudWatch 事件是将要终止 Auto Scaling 组中的一个实例。您可以配置 CloudWatch，使得此事件的目标是 Run Command 脚本，该脚本在实例终止前从实例上捕获日志文件。您还可在配置在 Auto Scaling 组中创建了新实例时执行的 Run Command 操作。例如，在 CloudWatch 接收实例已创建的事件时，Run Command 可以启用 Web 服务器角色或者在实例上安装软件。

- [为 Run Command 配置 CloudWatch Events \(p. 173\)](#)
- [将 Run Command 配置为 CloudWatch Events 目标 \(p. 174\)](#)

为 Run Command 配置 CloudWatch Events

您可以配置 CloudWatch Events 来向您通知 Run Command 状态更改，或者特定命令调用的状态更改。使用以下过程配置 CloudWatch Events 以发送有关 Run Command 的通知。

为 Run Command 配置 CloudWatch Events

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. 在左侧导航窗格中，选择 Events，然后选择 Create rule。
3. 在 Event Source 下，验证已选中 Event Pattern。
4. 在 Service Name 字段中，选择 EC2 Simple Systems Manager (SSM)

5. 在 Event Type 字段中，选择 Run Command。
6. 选择您要接收通知的详细类型和状态，然后选择 Add targets。
7. 在 Select target type 列表中，选择目标类型。有关不同目标类型的信息，请参阅对应的 AWS 帮助文档。
8. 选择 Configure details。
9. 指定规则详细信息，然后选择 Create rule。

将 Run Command 配置为 CloudWatch Events 目标

使用以下过程配置 Run Command 操作作为 CloudWatch 事件目标。

配置 Run Command 作为 CloudWatch 事件目标

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. 在左侧导航窗格中，选择 Events，然后选择创建新规则还是编辑现有规则。
3. 指定或验证规则的详细信息之后，选择 Add target。
4. 在 Select target type 列表中，选择 SSM Run Command。
5. 在 Document 列表中，选择 SSM 文档。该文档确定 Run Command 可在您实例上执行的操作类型。

Note

验证您选择的文档是否可在实例操作系统上运行。一些文档仅适用于 Windows 或 Linux 操作系统。有关 SSM 文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

6. 在 Target key 字段中，指定 InstanceIds 或 tag:[EC2_tag_name](#)。以下是使用 EC2 标签的 Target key 的一些示例：tag:production 和 tag:server-role。
7. 在 Target value(s) 字段中，如果在上一步选择了“InstanceIds”，则指定一个或多个实例 ID，以逗号分隔。如果您在上一步中选择了“tag:[EC2_tag_name](#)”，则指定一个或多个标签值。在键入值之后，例如 web-server 或 database，选择 Add。
8. 在 Configure parameter(s) 部分中，选择选项，然后完成由您的选项填充的所有字段。将鼠标悬停在文本上可获取有关选项的更多信息。有关文档的参数字段的更多信息，请查看 [使用 Systems Manager Run Command 运行命令 \(p. 181\)](#) 并选择适用于您文档的过程。
9. 在权限部分中，选择 Create a new role for this specific resource 可为 Run Command 创建带有所需实例配置文件角色的新角色。或者，选择 Use existing role。有关 Run Command 所需角色的更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。
10. 选择 Configure details 并完成向导。

为 Run Command 配置 Amazon SNS 通知

针对使用 Systems Manager Run Command 发送的命令，您可以将 Amazon Simple Notification Service (Amazon SNS) 配置为发送有关这些命令的状态的通知。Amazon SNS 协调并管理传输，或将通知发送到订阅客户端或终端节点。您可以在命令更改为新状态或特定状态（例如“失败”或“超时”）时接收通知。如果您将一条命令发送给多个实例，则可接收发送给特定实例的命令的每个副本的通知。每个副本称为一个调用。

Amazon SNS 能够以 HTTP 或 HTTPS POST 以及电子邮件 (SMTP、纯文本或 JSON 格式) 的方式传输通知，或者将通知作为消息发布到 Amazon Simple Queue Service (Amazon SQS) 队列。有关更多信息，请参阅 Amazon Simple Notification Service Developer Guide 中的 [什么是 Amazon SNS](#)。

为 Systems Manager 配置 Amazon SNS 通知

对于进入以下状态的命令，Run Command 支持发送相应的 Amazon SNS 通知。有关导致命令进入以下状态之一的条件的信息，请参阅 [了解命令状态 \(p. 189\)](#)。

- 正在进行
- 成功
- 已失败
- 超时
- 已取消

Note

使用 Run Command 发送的命令还报告“正在取消”和“待处理”状态。Amazon SNS 通知不会捕获这些状态。

如果您为 Amazon SNS 通知配置 Run Command，Amazon SNS 将发送包含以下信息的摘要消息：

字段	类型	描述
EventTime	字符串	触发事件的时间。由于 Amazon SNS 不保证消息传输顺序，因此时间戳很重要。示例：2016-04-26T13:15:30Z
DocumentName	字符串	用于运行此命令的 SSM 文档的名称。
CommandId	字符串	Run Command 在发送命令后生成的 ID。
ExpiresAfter	日期	如果达到此时间但命令尚未开始执行，则它将不会运行。
OutputS3BucketName	字符串	将命令执行响应存储到的 Amazon Simple Storage Service (Amazon S3) 存储桶。
OutputS3KeyPrefix	字符串	将命令执行响应存储到的存储桶中的 Amazon S3 目录路径。
RequestedDateTime	字符串	请求发送到此特定实例的日期和时间。
实例 ID	字符串	命令的目标实例。
状态	字符串	命令的命令状态。

如果您将一条命令发送给多个实例，则 Amazon SNS 可发送有关命令的每个副本或调用的消息，其中包含以下信息：

字段	类型	描述
EventTime	字符串	触发事件的时间。由于 Amazon SNS 不保证消息传输顺序，因此时间戳很重要。示例：2016-04-26T13:15:30Z
DocumentName	字符串	用于运行此命令的 Systems Manager 文档的名称。

字段	类型	描述
RequestedDateTime	字符串	请求发送到此特定实例的日期和时间。
CommandId	字符串	Run Command 在发送命令后生成的 ID。
实例 ID	字符串	命令的目标实例。
状态	字符串	此调用的命令状态。

要设置命令更改状态时的 Amazon SNS 通知，您必须完成以下任务。

主题

- [任务 1：为 Amazon SNS 通知创建 IAM 角色 \(p. 176\)](#)
- [任务 2：将 iam:PassRole 策略附加到您的 Amazon SNS 角色 \(p. 177\)](#)
- [任务 3：创建并订阅 Amazon SNS \(p. 177\)](#)
- [任务 4：发送可返回状态通知的命令 \(p. 177\)](#)

任务 1：为 Amazon SNS 通知创建 IAM 角色

使用以下过程为 Amazon SNS 通知创建 IAM 角色。此服务角色供 Systems Manager 用来触发 Amazon SNS 通知。

为 Amazon SNS 通知创建 IAM 服务角色

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择 Roles，然后选择 Create Role。
3. 在 Select type of trusted entity 页面上的 AWS Service 下，选择 EC2。
4. 在选择您的使用案例部分，选择 EC2，然后选择下一步：权限。
5. 在附加的权限策略页面中，搜索 AmazonSNSFullAccess 策略，选择它，然后选择下一步：审核。
6. 在 Review 页面上，在 Role name 框中键入名称，然后键入描述。
7. 选择 Create role。系统将让您返回到 Roles 页。
8. 在角色页面中，选择刚刚创建的角色以打开摘要页面。
9. 选择 Trust Relationships 选项卡，然后选择 Edit Trust Relationship。
10. 将 "ssm.amazonaws.com" 添加到现有策略，如以下代码段所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com",
        "Service": "ssm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Note

您必须在现有条目后添加逗号。"Service": "ec2.amazonaws.com"，否则 JSON 将不会验证。

11. 选择 Update Trust Policy。
12. 复制或记下 Role ARN。在发送配置为返回通知的命令时将指定此 ARN。
13. 使摘要页面保持打开状态。

任务 2：将 iam:PassRole 策略附加到您的 Amazon SNS 角色

要从 Amazon SNS 服务接收通知，您必须将 iam: PassRole 策略附加到任务 1 中创建的服务角色。

将 iam:PassRole 策略附加到您的 Amazon SNS 角色

1. 在刚刚创建的角色摘要页面中，选择权限选项卡。
2. 选择添加内联策略。
3. 在创建策略页面上，选择可视化编辑器选项卡。
4. 选择服务，然后选择 IAM。
5. 选择选择操作。
6. 在筛选操作文本框中，键入 PassRole，然后选择 PassRole 选项。
7. 选择资源。确保已选择 Specific，然后选择添加 ARN。
8. 在 Specify ARN for role 字段中，粘贴您在任务 1 结束时复制的 Amazon SNS 角色 ARN。系统会自动填充 Account 和 Role name with path 字段。
9. 选择 Add。
10. 选择查看策略。
11. 在查看策略页面上键入一个名称，然后选择创建策略。

任务 3：创建并订阅 Amazon SNS

Amazon SNS 主题是一种通信渠道，Run Command 使用它发送有关命令状态的通知。Amazon SNS 支持不同的通信协议，包括 HTTP/S、电子邮件和其他 AWS 服务，如 Amazon SQS。出于快速入门的目的，我们建议您先使用电子邮件协议。有关如何创建主题的信息，请参阅 Amazon Simple Notification Service Developer Guide 中的[创建主题](#)。

Note

创建主题后，复制或记下 Topic ARN。在发送配置为返回状态通知的命令时将指定此 ARN。

创建主题后，可指定终端节点订阅该主题。如果您选择电子邮件协议，终端节点即为您希望接收通知的电子邮件地址。有关如何订阅主题的更多信息，请参阅 Amazon Simple Notification Service Developer Guide 中的[订阅主题](#)。

Amazon SNS 从 AWS 通知向您指定的电子邮件地址发送确认电子邮件。打开这封电子邮件，然后选择确认订阅链接。

您将从 AWS 收到确认消息。Amazon SNS 现已配置为接收通知并将通知以电子邮件形式发送到您指定的电子邮件地址。

任务 4：发送可返回状态通知的命令

此部分介绍如何使用控制台或 AWS Command Line Interface (AWS CLI) 发送配置为返回状态通知的命令。

从控制台发送可返回通知的命令

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择运行命令。
4. 在命令文档列表中，选择一个 Systems Manager 文档。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. 在命令参数部分中，为必需的参数指定值。
7. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
8. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
9. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

10. 在 SNS Notifications 部分中，选择 Enable SNS notifications。
11. 在 IAM 角色字段中，键入或粘贴之前创建的 IAM 角色 ARN。
12. 在 SNS 主题字段中，键入或粘贴之前创建的 Amazon SNS ARN。
13. 在 Notify me on 字段中，选择要接收其通知的事件。
14. 在 Notify me for 字段中，选择接收发送给多个实例 (调用) 或命令摘要的命令的每个副本的通知。
15. 选择 Run。
16. 检查来自 Amazon SNS 的电子邮件消息并打开电子邮件。Amazon SNS 发送电子邮件需要花费几分钟的时间。

从 AWS CLI 发送为通知配置的命令

1. 打开 AWS CLI。

2. 指定以下命令中的参数。

```
aws ssm send-command --instance-ids "ID-1, ID-2" --document-name "name" --parameters
commands=date --service-role ServiceRole ARN --notification-config NotificationArn=SNS
ARN
```

例如

```
aws ssm send-command --instance-ids "i-12345678, i-34567890" --document-name "AWS-
RunPowerShellScript" --parameters commands=date --service-role arn:aws-cn:iam::
123456789012:myrole --notification-config NotificationArn=arn:aws-cn:sns:cn-
north-1:123456789012:test
```

3. 按 Enter。
4. 检查来自 Amazon SNS 的电子邮件消息并打开电子邮件。Amazon SNS 发送电子邮件需要花费几分钟的时间。

有关从命令行配置 Run Command 的更多信息，请参阅 [Amazon EC2 Systems Manager API Reference](#) 和 [Systems Manager AWS CLI 参考](#)。

基于实例标签限制 Run Command 访问

您可以通过创建一个 IAM 用户策略将命令执行进一步限制为特定实例，该用户策略包含一个条件，规定用户只能对使用特定 Amazon EC2 标签标记的实例运行命令。在以下示例中，用户可以通过使用任何实例 (Resource: arn:aws:ec2:*:*:instance/*) 上的任何 SSM 文档 (Resource: arn:aws:ssm:*:*:document/*) 使用 Run Command (Effect: Allow、Action: ssm:SendCommand)，条件是实例是 Finance WebServer (ssm:resourceTag/Finance: WebServer)。如果用户向未设置标签或具有除 Finance: WebServer 以外的任意标签的实例发送命令，则执行结果将显示 AccessDenied。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand"
      ],
      "Resource": [
        "arn:aws:ssm:*:*:document/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*"
      ],
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/Finance": [
            "WebServers"
          ]
        }
      }
    }
  ]
}
```

您可以创建支持用户对使用多个标签进行标记的实例运行命令的 IAM 策略。以下策略支持用户对具有两个标签的实例运行命令。如果用户向未使用这两个标签标记的实例发送命令，则执行结果将显示 `AccessDenied`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/tag_key1": [
            "tag_value1"
          ],
          "ssm:resourceTag/tag_key2": [
            "tag_value2"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand"
      ],
      "Resource": [
        "arn:aws:ssm:us-west-1::document/AWS-*",
        "arn:aws:ssm:us-east-2::document/AWS-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:UpdateInstanceInformation",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations",
        "ssm:GetDocument"
      ],
      "Resource": "*"
    }
  ]
}
```

您也可以创建支持用户对多个标记的实例组运行命令的 IAM 策略。以下策略支持用户对任一标记的实例组或两个标记的实例组运行命令。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/tag_key1": [
            "tag_value1"
          ]
        }
      }
    }
  ]
}
```

```
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:SendCommand"
    ],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "ssm:resourceTag/tag_key2": [
          "tag_value2"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:SendCommand"
    ],
    "Resource": [
      "arn:aws:ssm:us-west-1::document/AWS-*",
      "arn:aws:ssm:us-east-2::document/AWS-*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "ssm:UpdateInstanceInformation",
      "ssm:ListCommands",
      "ssm:ListCommandInvocations",
      "ssm:GetDocument"
    ],
    "Resource": "*"
  }
]
```

有关创建 IAM 用户策略的更多信息，请参阅 IAM User Guide 中的[托管策略与内联策略](#)。有关标记实例的更多信息，请参阅 Amazon EC2 User Guide for Linux Instances 的[标记您的 Amazon EC2 资源](#)（内容适用于 Windows 和 Linux 实例）。

使用 Systems Manager Run Command 运行命令

本部分包括有关如何从 Amazon EC2 控制台发送命令的信息，以及如何使用带有 EC2 标签的 Targets 参数将命令发送到实例队列。此部分还包括有关如何取消命令的信息。

有关使用 Windows PowerShell 发送命令的信息，请参阅[演练：将 AWS Tools for Windows PowerShell 与 Run Command 结合使用](#) (p. 194) 或[Tools for Windows PowerShell 参考](#)中的示例。有关如何使用 AWS CLI 发送命令的信息，请参阅[演练：将 AWS CLI 与 Run Command 结合使用](#) (p. 191) 或[SSM CLI 参考](#)中的示例。

Important

如果这是您第一次使用 Run Command，建议您对测试实例或未在生产环境中使用的实例执行命令。

内容

- [从控制台运行命令](#) (p. 182)

- [发送使用文档版本参数的命令 \(p. 184\)](#)
- [将命令发送到队列 \(p. 185\)](#)
- [通过脚本重启托管实例 \(p. 187\)](#)
- [取消命令 \(p. 188\)](#)

从控制台运行命令

您无需登录到每个实例就可以从控制台使用 Run Command 配置实例。此主题包括演示如何使用 Run Command 在实例上[更新 SSM 代理 \(p. 183\)](#)的示例。

开始前的准备工作

在使用 Run Command 发送命令之前，请验证您的实例是否符合 Systems Manager [要求 \(p. 6\)](#)。

使用 Run Command 发送命令 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 在命令文档列表中，选择一个 Systems Manager 文档。
4. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

5. 在命令参数部分中，为必需的参数指定值。
6. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
7. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
8. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

9. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅[Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

10. 选择 Run。

有关取消命令的信息，请参阅 [the section called “取消命令” \(p. 188\)](#)。

示例：更新 SSM 代理

您可使用 AWS-UpdateSSMAgent 文档来更新在 Windows 和 Linux 实例上运行的 Amazon EC2 SSM 代理。您可以更新到最新版本或降级到较旧版本。在运行命令时，系统将从 AWS 下载并安装需要的版本，然后卸载运行命令前存在的版本。如果此过程中出现错误，系统将回滚到命令运行之前服务器上的版本，并且命令状态将显示命令失败。

使用 Run Command 发送命令 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 在 Command document 列表中，选择 AWS-UpdateSSMAgent。
4. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

5. 在命令参数部分中，根据需要为以下参数指定值：
 - a. (可选) 对于版本，键入要安装的 SSM 代理的版本。您可以安装代理的[较旧版本](#)。如果您不指定版本，则服务将安装最新版本。
 - b. (可选) 对于 Allow Downgrade (允许降级)，选择 true (真) 以安装 SSM 代理的早期版本。如果选择此选项，则必须指定[较早](#)的版本号。选择 false 以仅安装此服务的最新版本。
6. 在 Other parameters 中：
 - 在 Comment 框中，键入有关此命令的信息。
 - 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。
7. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
8. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

- 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

- 选择 Run。

发送使用文档版本参数的命令

您可以使用 document-version 参数指定在运行命令时使用 SSM 文档的哪个版本。您可以为此参数指定以下选项之一：

- \$DEFAULT
- \$LATEST
- 版本号

如果您使用 AWS CLI 运行命令，则必须使用反斜杠转义前两个选项。如果您指定了版本号，则不需要使用反斜杠。例如：

```
--document-version "$DEFAULT"
```

```
--document-version "$LATEST"
```

```
--document-version "3"
```

使用以下过程，通过 AWS CLI 运行使用 document-version 参数的命令。

使用 AWS CLI 运行命令

- 运行以下命令指定您的凭证和区域。

```
aws configure
```

- 系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

*##*代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 *us-east-2* 表示 US East (Ohio) Region。有关受支持 *##* 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

- 列出所有可用文档

此命令将基于 IAM 权限列出您的账户可用的所有文档。此命令将返回 Linux 和 Windows 文档的列表。

```
aws ssm list-documents
```

4. 使用以下命令查看文档的不同版本。

```
aws ssm list-document-versions --name "document name"
```

5. 通过以下命令运行使用 SSM 文档版本的命令。

```
aws ssm send-command --document-name "AWS-RunShellScript" --parameters commands="echo  
Hello",executionTimeout=3600 --instance-ids instance-ID --endpoint-url "https://us-  
east-2.amazonaws.com" --region "us-east-2" --document-version "\$DEFAULT, \$LATEST, or  
a version number"
```

将命令发送到队列

您可以使用 `targets` 参数向数十个、数百个或数千个实例发送命令 (Amazon EC2 控制台中的 `Select Targets by Specifying a Tag` 选项)。`targets` 参数支持基于您为实例指定的 Amazon EC2 标签 `Key, Value` 组合。当您运行该命令时，系统会找到并尝试在匹配指定标签的所有实例上运行命令。有关 Amazon EC2 标签的更多信息，请参阅 Amazon EC2 用户指南 中的 [标记 Amazon EC2 资源](#) (内容适用于 Windows 和 Linux 实例)。

Note

您也可以使用 `targets` 参数将特定实例 ID 的列表设为目标，如下一部分中所述。

为控制数百个或数千个实例的命令执行，Run Command 还包含一些参数，用于限制同时处理请求的实例数量以及终止命令前其引发的错误数量。

内容

- [针对多个实例 \(p. 185\)](#)
- [使用并发控件 \(p. 187\)](#)
- [使用错误控件 \(p. 187\)](#)

针对多个实例

您可以通过指定 Amazon EC2 标签或实例 ID 运行命令并将实例设为目标。在 AWS CLI 中，`targets` 参数使用以下语法：

示例 1：将标签设为目标

```
aws ssm send-command --document-name name --targets Key=tag:tag_name,Values=tag_value [...]
```

Note

本部分的示例命令使用 [...] 进行截断。

示例 2：将实例 ID 设为目标

```
aws ssm send-command --document-name name --targets Key=instanceids,Values=ID1, ID2, ID3  
[...]
```

如果您使用名为 `Key` 的 `Environment`，以及 `Values`、`Development`、`Test` 和 `Pre-production` 的 标记不同环境的实例，则可以使用采用以下语法的 参数，向其中一个环境的所有实例发送命令。`targets`

```
aws ssm send-command --document-name name --targets Key=tag:Environment,Values=Development [...]
```

通过添加到 Values 列表，您可以将其他环境中的其他实例设为目标。使用逗号分隔项目。

```
aws ssm send-command --document-name name --targets  
Key=tag:Environment,Values=Development,Test,Pre-production [...]
```

示例：使用多个 Key 条件细化您的目标。

通过包括多个 Key 条件，您可以细化您的命令的目标数。如果包括多个 Key 条件，系统会将符合所有条件的实例设为目标。以下命令会将标记为财务部门和标记为数据库服务器角色的所有实例设为目标。

```
aws ssm send-command --document-name name --targets Key=tag:Department,Values=Finance  
Key=tag:ServerRole,Values=Database [...]
```

示例：使用多个 Key 和 Value 条件

对上一个示例进行扩展，您可以通过在 Values 条件中包括其他项目来将多个部门和多个服务器角色设为目标。

```
aws ssm send-command --document-name name --targets  
Key=tag:Department,Values=Finance,Marketing Key=tag:ServerRole,Values=WebServer,Database [...]
```

示例：针对使用多个 Values 条件标记的实例

如果您使用名为 Department 的 Key，以及 Sales 和 Finance 的 Values 标记不同环境的实例，则可以使用采用以下语法的 targets 参数，向这些环境的所有实例发送命令。

```
aws ssm send-command --document-name name --targets Key=tag:Department,Values=Sales,Finance [...]
```

Note

您可以指定最大值为 5 个密钥，每个密钥 5 个值。

如果某个标签密钥 (标签名称) 或某个标签值包含空格，则必须将该标签密钥或该值用引号引起来，如以下示例所示。

示例 1：Value 标签中的空格。

```
aws ssm send-command --document-name name --targets Key=tag:OS,Values="Windows Server 2016 Nano" [...]
```

示例 2：tag 密钥和 Value 中的空格。

```
aws ssm send-command --document-name name --targets Key="tag:Operating System",Values="Windows Server 2016 Nano" [...]
```

示例 3：Values 的列表中一个项目中的空格。

```
aws ssm send-command --document-name name --targets  
Key=tag:Department,Values="Sales", "Finance", "Systems Mgmt" [...]
```

使用并发控件

您可以使用 `max-concurrency` 参数 (Amazon EC2 控制台中的 Execute on (执行位置) 字段) 控制同时运行命令的服务器的数量。您可以指定绝对数量的实例 (如 10)，也可以指定目标集百分比 (如 10%)。队列系统将该命令传输到单个实例，等到初始调用完成之后，再将命令发送到两个或更多实例。系统以指数增长方式将命令发送到更多实例，直到达到 `max-concurrency` 值。`max-concurrency` 值默认为 50。下列示例介绍如何为 `max-concurrency` 参数指定值：

```
aws ssm send-command --document-name name --max-concurrency 10 --targets  
Key=tag:Environment,Values=Development [...]
```

```
aws ssm send-command --document-name name --max-concurrency 10% --targets  
Key=tag:Department,Values=Finance,Marketing Key=tag:ServerRole,Values=WebServer,Database  
[...]
```

使用错误控件

通过使用 `max-errors` 参数 (Amazon EC2 控制台中的 Stop after __ errors 字段) 设置错误限制，您还可以控制向数百个或数千个实例发送的命令的执行。该参数指定系统停止向其他实例发送命令之前所允许的错误数。您可以指定绝对数量的错误 (如 10)，也可以指定目标集百分比 (如 10%)。例如，如果您指定 3，系统将在收到第四个错误时停止发送命令。如果您指定 0，则系统会在返回第一个错误结果后停止向其他实例发送命令。如果您向 50 个实例发送命令并将 `max-errors` 设置为 10%，则系统会在收到第六个错误时停止向其他实例发送命令。

当达到 `max-errors` 时，允许完成已经运行命令的调用，但是其中一些调用也可能失败。如果您需要确保失败的调用数不超过 `max-errors`，请将 `max-concurrency` 设置为 1，以便一次进行一个调用。`max-errors` 默认为 0。下列示例介绍了如何为 `max-errors` 参数指定值：

```
aws ssm send-command --document-name name --max-errors 10 --targets  
Key=tag:Database,Values=Development [...]
```

```
aws ssm send-command --document-name name --max-errors 10% --targets  
Key=tag:Environment,Values=Development [...]
```

```
aws ssm send-command --document-name name --max-concurrency 1 --max-errors 1 --targets  
Key=tag:Environment,Values=Production [...]
```

通过脚本重启托管实例

如果您使用 Run Command 运行的脚本用于重启托管实例，则您必须在脚本中指定一个退出代码。如果您使用其他一些机制尝试从脚本重启实例，则即使重启是脚本的最后一步，脚本执行状态也可能无法正确更新。对于 Windows 托管实例，您需在脚本中指定 `exit 3010`。对于 Linux 托管实例，您需指定 `exit 194`。退出代码用于指示 SSM 代理 重启托管实例，然后在重启完成后重新启动脚本。在重启开始之前，SSM 代理会通知云中的 Systems Manager 服务，通信将在服务器重启期间中断。

创建幂等脚本

在开发用于重启托管实例的脚本时，使脚本具有幂等性，以便脚本执行在重启后从中断的位置继续进行。

以下是多次重启实例的幂等脚本的概要示例。

```
$name = Get current computer name  
If ($name -ne $desiredName)  
{
```

```
Rename computer
exit 3010
}

$domain = Get current domain name
If ($domain -ne $desiredDomain)
{
    Join domain
    exit 3010
}

If (desired package not installed)
{
    Install package
    exit 3010
}
```

以下脚本示例使用退出代码来重新启动实例。Windows 示例在实例上安装 Hyperv-V 应用程序，然后重新启动该实例。Linux 示例在 Amazon Linux 上安装程序包更新，然后重新启动该实例。

Windows 示例

```
$hyperV = Get-WindowsFeature -Name Hyper-V
if(-not $hyperV.Installed)
{
    # Install Hyper-V and then send a reboot request to SSM Agent.
    Install-WindowsFeature -Name Hyper-V -IncludeManagementTools
    exit 3010
}
```

Amazon Linux 示例

```
#!/bin/bash
yum -y update
needs-restarting -r
if [ $? -eq 1 ]
then
    exit 194
else
    exit 0
fi
```

取消命令

只要服务指明命令处于 Pending 或 Executing 状态，您就可以尝试取消该命令。但是，即使命令仍处于其中某种状态，我们也无法保证该命令将终止并且基础流程将停止。

使用控制台取消命令 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择要取消的命令调用。
4. 选择取消命令。

使用 AWS CLI 取消命令

使用以下命令。

```
aws ssm cancel-command --command-id "command ID" --instance-ids "instance ID"
```

有关已取消命令的状态的信息，请参阅[了解命令状态 \(p. 189\)](#)。

了解命令状态

Systems Manager Run Command 报告关于处理过程中命令经历的不同状态以及处理该命令的每个实例的详细状态信息。您可以使用以下方法监视命令状态。

- 在 Amazon EC2 控制台的 Run Command 页面上，单击 Refresh 图标。
- 使用 AWS CLI 调用 [list-commands](#) 或 [list-command-invocations](#)。或者，使用 AWS Tools for Windows PowerShell 调用 [Get-SSMCommand](#) 或 [Get-SSMCommandInvocation](#)。
- 将 CloudWatch Events 配置为记录状态更改。
- 将 Amazon SNS 配置为发送所有状态更改或特定状态（例如“失败”或“超时”）的通知。

Run Command 状态

Run Command 报告以下三个区域的状态详情：插件、调用和总体命令状态。插件是在您命令的 Systems Manager 文档中定义的代码执行块。AWS-* 文档仅包含一个插件，但是您可以自行创建使用多个插件的文档。有关插件的更多信息，请参阅 [SSM 文档插件参考 \(p. 312\)](#)。

在将一条命令同时发送给多个实例时，针对每个实例的命令的每个副本均为一个命令调用。例如，如果您使用 AWS-RunShellScript 文档并将一个 ifconfig 命令发送到 20 个实例，则该命令具有 20 个调用。每个命令调用都会单独报告状态。给定命令调用的插件也会单独报告状态。

最后，Run Command 包含适用于所有插件和调用的聚合命令状态。聚合命令状态可能不同于插件或调用报告的状态，如下表所述。

Note

如果您使用 `max-concurrency` 或 `max-errors` 参数对大量实例运行命令，则命令状态会反映这些参数施加的限制，如下表所述。有关这些参数的更多信息，请参阅 [将命令发送到队列 \(p. 185\)](#)。

命令插件和调用的详细状态

状态	详细信息
Pending	实例上的代理尚未收到命令。如果代理在达到超时(秒)参数指定的值之前还未收到命令，则状态会变为 <code>Delivery Timed Out</code> 。
正在进行	代理已收到命令，或者命名已开始实例上执行。根据所有命令插件的结果，状态会变为 <code>Success</code> 、 <code>Failed</code> 或 <code>Execution Timed Out</code> 。如果代理在实例上不可用，则命令状态会显示 <code>In Progress</code> ，直至代理再次可用。状态将变为最终状态。
延迟	系统尝试向实例发送命令，但未成功。系统将重试。

状态	详细信息
成功	<p>命令由实例上的 SSM 代理接收，并返回一个为零的退出代码。此状态并不意味着命令在实例上处理成功。这是最终状态。</p> <p>Note</p> <p>要解决错误或获取有关命令执行的更多信息，请通过返回适当的退出代码 (针对命令失败的非零退出代码) 来发送一个处理错误或异常的命令。</p>
传输超时	<p>命令未在传输超时过期之前传输到实例。传输超时不计入父命令的 <code>max-errors</code> 限制，但是有助于区别父命令的状态是 <code>Success</code> 还是 <code>Incomplete</code>。这是最终状态。</p>
执行超时	<p>命令执行在实例上开始，但未在执行超时过期之前完成。执行超时不计入父命令的 <code>max-errors</code> 限制。这是最终状态。</p>
已失败	<p>实例上的命令失败了。对于插件，这表示结果代码不为零。对于命令调用，这表示一个或多个插件的结果代码不为零。调用失败不计入父命令的 <code>max-errors</code> 限制。这是最终状态。</p>
已取消	<p>命令在完成之前终止。这是最终状态。</p>
无法传输	<p>命令无法传输到实例。实例可能不存在或可能未响应。无法传输的调用不计入父命令的 <code>max-errors</code> 限制，而且无法用于区别父命令的状态是 <code>Success</code> 还是 <code>Incomplete</code>。这是最终状态。</p>
已终止	<p>父命令超过其 <code>max-errors</code> 限制且系统取消了后续命令调用。这是最终状态。</p>

命令的详细状态

状态	详细信息
Pending	<p>任何实例上的代理都尚未收到命令。</p>
正在进行	<p>命令已发送到至少一个实例，但是在所有实例上都未达到最终状态。</p>
延迟	<p>系统尝试向实例发送命令，但未成功。系统将重试。</p>
成功	<p>命令由所有指定或目标实例上的 SSM 代理接收，并返回一个为零的退出代码。所有命令调用都已达到最终状态，且未超过值 <code>max-errors</code>。此状态并不意味着命令在所有指定或目标实例上处理成功。这是最终状态。</p> <p>Note</p> <p>要解决错误或获取有关命令执行的更多信息，请通过返回适当的退出代码 (针对命令</p>

状态	详细信息
	失败的非零退出代码) 来发送一个处理错误或异常的命令。
传输超时	命令未在传输超时过期之前传输到实例。max-errors 值或更多命令调用显示 Delivery Timed Out 状态。这是最终状态。
执行超时	命令执行在实例上开始，但未在执行超时过期之前完成。max-errors 值或更多命令调用显示 Execution Timed Out 状态。这是最终状态。
已失败	实例上的命令失败了。max-errors 值或更多命令调用显示 Failed 状态。这是最终状态。
未完成	命令尝试在所有实例上执行，且一个或多个调用没有 Success 值。不过，调用失败的次数不足以使状态变为 Failed。这是最终状态。
已取消	命令在完成之前终止。这是最终状态。
数量超出限制	命令定位的实例数量超出了待处理调用的账户限制。系统在实例上执行命令之前取消了命令。这是最终状态。

Run Command 演练

此部分中的演练向您演示如何使用 AWS Command Line Interface 或 AWS Tools for Windows PowerShell 通过 Run Command 运行命令。

内容

- [演练：将 AWS CLI 与 Run Command 结合使用 \(p. 191\)](#)
- [演练：将 AWS Tools for Windows PowerShell 与 Run Command 结合使用 \(p. 194\)](#)

您还可以查看以下参考中的示例命令。

- [Systems Manager AWS CLI 参考](#)
- [Systems Manager AWS Tools for Windows PowerShell 参考](#)

演练：将 AWS CLI 与 Run Command 结合使用

以下示例演练说明如何使用 AWS CLI 查看有关命令和命令参数的信息、如何运行命令以及如何查看这些命令的状态。

Important

仅允许受信任的管理员使用本主题中所示的 Systems Manager 预配置文档。在 Systems Manager 文档中指定的命令和脚本需要管理权限才能在您的实例上运行。如果用户有权运行任何预定义的 Systems Manager 文档（任何以 AWS 开头的文档），则该用户也具有实例的管理员访问权限。对于所有其他用户，您应创建限制性文档并与特定用户共享这些文档。有关限制对 Run Command 的访问权限的更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

主题

- [步骤 1：入门 \(p. 192\)](#)

- [步骤 2：运行 Shell 脚本 \(p. 192\)](#)
- [步骤 3：使用 AWS-RunShellScript 文档发送命令 - 示例 1 \(p. 193\)](#)
- [步骤 4：使用 AWS-RunShellScript 文档发送命令 - 示例 2 \(p. 193\)](#)
- [其他示例 \(p. 193\)](#)

步骤 1：入门

您必须具有要配置的实例的管理员权限或必须已获得 IAM 中的适当权限。另请注意，此示例将使用 US East (Ohio) Region (us-east-2)。Run Command 当前在 Amazon Web Services General Reference 的 [AWS Systems Manager](#) 中列出的 AWS 区域内可用。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

使用 AWS CLI 运行命令

1. 运行以下命令指定您的凭证和区域。

```
aws configure
```

2. 系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region-id
Default output format [None]: ENTER
```

*##*代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 us-east-2 表示 US East (Ohio) Region。有关受支持 *##* 值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

3. 列出所有可用文档

此命令将基于 IAM 权限列出您的账户可用的所有文档。此命令将返回 Linux 和 Windows 文档的列表。

```
aws ssm list-documents
```

4. 验证实例是否准备好接收命令

以下命令的输出将显示实例是否处于联机状态。

```
aws ssm describe-instance-information --output text --query
"InstanceInformationList[*]"
```

5. 使用以下命令查看有关特定实例的详细信息。

Note

要运行本演练中的命令，您必须替换实例和命令 ID。命令 ID 将作为 send-command 的响应返回。可从 Amazon EC2 控制台获得实例 ID。

```
aws ssm describe-instance-information --instance-information-filter-list
key=InstanceIds,valueSet=instance ID
```

步骤 2：运行 Shell 脚本

利用 Run Command 和 AWS-RunShellScript 文档，您可以在 EC2 实例上运行任何命令或脚本，就像您已在本地登录一样。

查看说明和可用参数

- 使用以下命令查看 Systems Manager JSON 文档的描述。

```
aws ssm describe-document --name "AWS-RunShellScript" --query  
"[Document.Name,Document.Description]"
```

- 使用以下命令查看可用参数和有关这些参数的详细信息。

```
aws ssm describe-document --name "AWS-RunShellScript" --query "Document.Parameters[*]"
```

步骤 3：使用 AWS-RunShellScript 文档发送命令 - 示例 1

使用以下命令获取实例的 IP 信息。

```
aws ssm send-command --instance-ids "instance ID" --document-name "AWS-RunShellScript" --  
comment "IP config" --parameters commands=ifconfig --output text
```

使用响应数据获取命令信息

以下命令使用从上一步命令返回的命令 ID 来获取命令执行的详细信息和响应数据。如果命令已完成，系统将返回响应数据。如果命令执行显示“Pending”，您将需要再次运行此命令来查看响应数据。

```
aws ssm list-command-invocations --command-id $sh_command_id --details
```

步骤 4：使用 AWS-RunShellScript 文档发送命令 - 示例 2

以下命令显示运行命令的默认用户账户。

```
sh_command_id=$(aws ssm send-command --instance-ids "instance ID" --document-name  
"AWS-RunShellScript" --comment "Demo run shell script on Linux Instance" --parameters  
commands=whoami --output text --query "Command.CommandId")
```

获取命令状态

以下命令使用命令 ID 获取实例上的命令执行的状态。此示例使用上一步命令中返回的命令 ID。

```
aws ssm list-commands --command-id "command_ID"
```

获取命令详细信息

以下命令使用来自上一步命令的命令 ID 来获取每个实例的命令执行的状态。

```
aws ssm list-command-invocations --command-id "command_ID" --details
```

获取带有特定实例的响应数据的命令信息

以下命令返回特定实例的原始 aws ssm send-command 的输出。

```
aws ssm list-command-invocations --instance-id instance ID --command-id "command_ID" --  
details
```

其他示例

以下命令返回在实例上运行的 Python 的版本。

```
sh_command_id=$(aws ssm send-command --instance-ids "instance ID" --document-name
"AWS-RunShellScript" --comment "Demo run shell script on Linux Instances" --
parameters commands='python -V' --output text --query "Command.CommandId") sh -c
'aws ssm list-command-invocations --command-id "$sh_command_id" --details --query
"CommandInvocations[].CommandPlugins[].{Status:Status,Output:Output}"'
```

以下命令使用 Run Command 运行 Python 脚本。

```
sh_command_id=$(aws ssm send-command --instance-ids "instance ID" --document-name
"AWS-RunShellScript" --comment "Demo run shell script on Linux Instances" --
parameters '{"commands":["#!/usr/bin/python","print \"Hello world from python\\\""]}'
--output text --query "Command.CommandId") sh -c 'aws ssm list-command-invocations --
command-id "$sh_command_id" --details --query "CommandInvocations[].CommandPlugins[.
{Status:Status,Output:Output}]"'
```

演练：将 AWS Tools for Windows PowerShell 与 Run Command 结合使用

以下示例说明如何使用 Tools for Windows PowerShell 查看有关命令和命令参数的信息、如何运行命令以及如何查看这些命令的状态。本演练为每个预定义的 Systems Manager 文档包含了一个示例。

Important

仅允许受信任的管理员使用本主题中所示的 Systems Manager 预配置文档。在 Systems Manager 文档中指定的命令和脚本需要管理权限才能在您的实例上运行。如果用户有权运行任何预定义的 Systems Manager 文档（任何以 AWS 开头的文档），则该用户也具有实例的管理员访问权限。对于所有其他用户，您应创建限制性文档并与特定用户共享这些文档。有关限制对 Run Command 的访问权限的更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

主题

- [配置 AWS Tools for Windows PowerShell 会话设置 \(p. 194\)](#)
- [列出所有可用文档 \(p. 195\)](#)
- [运行 PowerShell 命令或脚本 \(p. 195\)](#)
- [使用 AWS-InstallApplication 文档安装应用程序 \(p. 196\)](#)
- [使用 AWS-InstallPowerShellModule JSON 文档安装 PowerShell 模块 \(p. 197\)](#)
- [使用 AWS-JoinDirectoryServiceDomain JSON 文档将实例加入域 \(p. 197\)](#)
- [使用 AWS-ConfigureCloudWatch 文档将 Windows 指标发送到 Amazon CloudWatch \(p. 198\)](#)
- [使用 AWS-ConfigureWindowsUpdate 文档启用/禁用 Windows 自动更新 \(p. 199\)](#)
- [使用 AWS-UpdateEC2Config 文档更新 EC2Config \(p. 200\)](#)
- [使用 Run Command 管理 Windows 更新 \(p. 201\)](#)

配置 AWS Tools for Windows PowerShell 会话设置

在本地计算机上打开 AWS Tools for Windows PowerShell 并运行以下命令来指定凭证。您必须具有要配置的实例的管理员权限或必须已获得 IAM 中的适当权限。有关更多信息，请参阅[Systems Manager 先决条件 \(p. 6\)](#)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

执行以下命令为 PowerShell 会话设置区域。本示例使用 US East (Ohio) Region (us-east-2)。Run Command 当前在 Amazon Web Services General Reference 的 [AWS Systems Manager](#) 中列出的 AWS 区域内可用。

```
Set-DefaultAWSRegion -Region us-east-2
```

列出所有可用文档

此命令列出对您的账户可用的所有文档：

```
Get-SSMDocumentList
```

运行 PowerShell 命令或脚本

利用 Run Command 和 AWS-RunPowerShell 文档，您可使用远程桌面在 EC2 实例上运行任何命令或脚本，就像您已登录到该实例一样。您可以发出命令或键入本地脚本的路径以运行命令。

Note

有关在使用 Run Command 调用脚本时重启服务器和实例的信息，请参阅[通过脚本重启托管实例 \(p. 187\)](#)。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-RunPowerShellScript"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-RunPowerShellScript" | select -ExpandProperty  
Parameters
```

使用 AWS-RunPowerShellScript 文档发送命令

以下命令在两个实例上显示 C:\Users 目录的内容和 C:\ 目录的内容。

```
$runPSCmd=Send-SSMCommand -InstanceId @('Instance-ID', 'Instance-ID') -DocumentName  
AWS-RunPowerShellScript -Comment 'Demo AWS-RunPowerShellScript with two instances' -  
Parameter @{commands}=('@(dir C:\Users', 'dir C:\'))
```

获取命令请求详细信息

以下命令使用命令 ID 获取两个实例上的命令执行的状态。此示例使用上一个命令中返回的命令 ID。

```
Get-SSMCommand -CommandId $runPSCmd.CommandId
```

此示例中命令的状态可以是 Success、Pending 或 InProgress。

获取每个实例的命令信息

以下命令使用来自上一命令的命令 ID 来获取每个实例的命令执行的状态。

```
Get-SSMCommandInvocation -CommandId $runPSCmd.CommandId
```

获取带有特定实例的响应数据的命令信息

以下命令返回特定实例的原始 Send-SSMCommand 的输出。

```
Get-SSMCommandInvocation -CommandId $runPSCommand.CommandId -Details $true -  
InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

取消命令

以下命令取消 AWS-RunPowerShellScript 文档的 Send-SSMCommand。

```
$cancelCommandResponse=Send-SSMCommand -InstanceId @('Instance-ID','Instance-ID') -  
DocumentName AWS-RunPowerShellScript -Comment 'Demo AWS-RunPowerShellScript with two  
instances' -Parameter @{'commands'='Start-Sleep -Seconds 120; dir C:\'}  
Stop-SSMCommand -CommandId $cancelCommandResponse.CommandId Get-SSMCommand -CommandId  
$cancelCommandResponse.CommandId
```

查看命令状态

以下命令检查 Cancel 命令的状态

```
Get-SSMCommand -CommandId $cancelCommandResponse.CommandId
```

使用 AWS-InstallApplication 文档安装应用程序

利用 Run Command 和 AWS-InstallApplication 文档，您可以在实例上安装、修复或卸载应用程序。该命令需要 MSI 的路径或地址。

Note

有关在使用 Run Command 调用脚本时重启服务器和实例的信息，请参阅[通过脚本重启托管实例 \(p. 187\)](#)。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-InstallApplication"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-InstallApplication" | select -ExpandProperty  
Parameters
```

使用 AWS-InstallApplication 文档发送命令

以下命令将自动在您的实例上安装 Python 版本，并将输出记录在 C: 驱动器上的本地文本文件中。

```
$installAppCommand=Send-SSMCommand -InstanceId Instance-ID -DocumentName AWS-  
InstallApplication -Parameter @{'source'='https://www.python.org/ftp/python/2.7.9/  
python-2.7.9.msi'; 'parameters'='/norestart /quiet /log c:\pythoninstall.txt'}
```

获取每个实例的命令信息

以下命令使用命令 ID 获取命令执行的状态

```
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true
```

获取带有特定实例的响应数据的命令信息

以下命令返回 Python 安装的结果。

```
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true -  
InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

使用 AWS-InstallPowerShellModule JSON 文档安装 PowerShell 模块

您可使用 Run Command 在 EC2 实例上安装 PowerShell 模块。有关 PowerShell 模块的更多信息，请参阅[Windows PowerShell 模块](#)。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-InstallPowerShellModule"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-InstallPowerShellModule" | select -ExpandProperty  
Parameters
```

安装 PowerShell 模块

以下命令下载 EZOut.zip 文件，安装该文件，然后运行另一个命令来安装 XPS Viewer。最后，此命令的输出将上传到一个名为 demo-ssm-output-bucket 的 Amazon S3 存储桶中。

```
$installPSCommand=Send-SSMCommand -InstanceId Instance-ID -DocumentName AWS-  
InstallPowerShellModule -Parameter @{ 'source'='https://gallery.technet.microsoft.com/  
EZOut-33ae0fb7/file/110351/1/EZOut.zip'; 'commands'=@('Add-WindowsFeature -name XPS-Viewer -  
restart')} -OutputS3BucketName demo-ssm-output-bucket
```

获取每个实例的命令信息

以下命令使用命令 ID 获取命令执行的状态。

```
Get-SSMCommandInvocation -CommandId $installPSCommand.CommandId -Details $true
```

获取具有实例的响应数据的命令信息

以下命令返回特定命令 ID 的原始 Send-SSMCommand 的输出。

```
Get-SSMCommandInvocation -CommandId $installPSCommand.CommandId -Details $true | select -  
ExpandProperty CommandPlugins
```

使用 AWS-JoinDirectoryServiceDomain JSON 文档将实例加入域

借助 Run Command，您可以将实例快速加入 AWS Directory Service 域。在执行此命令前，您必须[创建目录](#)。还建议您了解有关 AWS Directory Service 的更多信息。有关更多信息，请参阅[什么是 AWS Directory Service ?](#)。

目前，您只能将一个示例加入域。无法从域中删除实例。

Note

有关在使用 Run Command 调用脚本时重启服务器和实例的信息，请参阅[通过脚本重启托管实例 \(p. 187\)](#)。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-JoinDirectoryServiceDomain"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-JoinDirectoryServiceDomain" | select -ExpandProperty Parameters
```

将实例加入域

以下命令将实例加入给定的 AWS Directory Service 域，并将任何生成的输出上传到 Amazon S3 存储桶中。

```
$domainJoinCommand=Send-SSMCommand -InstanceId Instance-ID -DocumentName  
AWS-JoinDirectoryServiceDomain -Parameter @{ 'directoryId'='d-9067386b64';  
'directoryName'='ssm.test.amazon.com'; 'dnsIpAddresses'=@('172.31.38.48',  
'172.31.55.243')} -OutputS3BucketName demo-ssm-output-bucket
```

获取每个实例的命令信息

以下命令使用命令 ID 获取命令执行的状态。

```
Get-SSMCommandInvocation -CommandId $domainJoinCommand.CommandId -Details $true
```

获取具有实例的响应数据的命令信息

此命令返回特定命令 ID 的原始 Send-SSMCommand 的输出。

```
Get-SSMCommandInvocation -CommandId $domainJoinCommand.CommandId -Details $true | select -  
ExpandProperty CommandPlugins
```

使用 AWS-ConfigureCloudWatch 文档将 Windows 指标发送到 Amazon CloudWatch

您可以将应用程序、系统、安全和 Windows 事件跟踪 (ETW) 日志中的 Windows Server 消息发送到 Amazon CloudWatch Logs。在首次启用日志记录时，Systems Manager 会发送从您开始上传该应用程序、系统、安全和 ETW 日志时起 1 分钟内生成的所有日志。其中不包括在此时间之前产生的日志。如果您禁用日志记录并在以后再次启用日志记录，则 Systems Manager 会从其上次停止的时间继续发送日志。对于任何自定义日志文件和 Internet Information Services (IIS) 日志，Systems Manager 会从头读取日志文件。此外，Systems Manager 还可以将性能计数器数据发送到 Amazon CloudWatch。

如果您先前在 EC2Config 中启用了 CloudWatch 集成，那么 Systems Manager 设置会覆盖实例本地在 C:\Program Files\Amazon\EC2ConfigService\Settings\AWS.EC2.Windows.CloudWatch.json 文件中存储的任何设置。有关使用 EC2Config 在一个实例上管理性能计数器和日志的更多信息，请参阅[将性能计数器发送到 CloudWatch 并将日志发送到 CloudWatch 日志](#)。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch" | select -ExpandProperty Parameters
```


将应用程序日志发送到 CloudWatch

以下命令配置实例并将 Windows 应用程序日志移至 CloudWatch。

```
$cloudWatchCommand=Send-SSMCommand -InstanceID Instance-ID -DocumentName  
'AWS-ConfigureCloudWatch' -Parameter @{'properties'='{ "engineConfiguration":  
{ "PollInterval": "00:00:15", "Components": [{ "Id": "ApplicationEventLog",  
"FullName": "AWS.EC2.Windows.CloudWatch.EventLog.EventLogInputComponent, AWS.EC2.Windows.CloudWatch",  
"Parameters": { "LogName": "Application", "Levels": "7" } }, { "Id": "CloudWatch",  
"FullName": "AWS.EC2.Windows.CloudWatch.CloudWatchLogsOutput, AWS.EC2.Windows.CloudWatch",  
"Parameters": { "Region": "us-east-2", "LogGroup": "My-Log-Group",  
"LogStream": "i-1234567890abcdef0" } } ], "Flows": { "Flows":  
[ "ApplicationEventLog, CloudWatch" ] } }' }
```

获取每个实例的命令信息

以下命令使用命令 ID 获取命令执行的状态。

```
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true
```

获取带有特定实例的响应数据的命令信息

以下命令返回 Amazon CloudWatch 配置的结果。

```
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true -  
InstanceId Instance-ID | select -ExpandProperty CommandPlugins
```

使用 AWS-ConfigureCloudWatch 文档将性能计数器发送到 CloudWatch

以下演示命令将性能计数器数据上传到 CloudWatch。有关更多信息，请参阅 [Amazon CloudWatch 文档](#)。

```
$cloudWatchMetricsCommand=Send-SSMCommand -InstanceID Instance-ID -DocumentName  
'AWS-ConfigureCloudWatch' -Parameter @{'properties'='{ "engineConfiguration":  
{ "PollInterval": "00:00:15", "Components": [{ "Id": "PerformanceCounter",  
"FullName": "AWS.EC2.Windows.CloudWatch.PerformanceCounterComponent.PerformanceCounterInputComponent, AW  
"Parameters": { "CategoryName": "Memory", "CounterName": "Available  
MBytes", "InstanceName": "", "MetricName": "AvailableMemory",  
"Unit": "Megabytes", "DimensionName": "", "DimensionValue": "" } }, { "Id": "CloudWatch",  
"FullName": "AWS.EC2.Windows.CloudWatch.CloudWatch.CloudWatchOutputComponent, AWS.EC2.Windows.CloudWatch  
"Parameters": { "AccessKey": "", "SecretKey": "", "Region": "us-east-2", "NameSpace": "Windows-  
Default" } } ], "Flows": { "Flows": [ "PerformanceCounter, CloudWatch" ] } }' }
```

使用 AWS-ConfigureWindowsUpdate 文档启用/禁用 Windows 自动更新

利用 Run Command 和 AWS-ConfigureWindowsUpdate 文档，您可以在 Windows 实例上启用或禁用自动 Windows 更新。此命令将 Windows 更新代理配置为在您指定的日期和时间下载并安装 Windows 更新。如果更新需要重启，计算机将在安装更新 15 分钟后自动重启。利用此命令，您还可以将 Windows 更新配置为检查更新但不安装更新。AWS-ConfigureWindowsUpdate 文档可与 Windows Server 2008、2008 R2、2012、2012 R2 和 2016 兼容。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-ConfigureWindowsUpdate"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-ConfigureWindowsUpdate" | select -ExpandProperty  
Parameters
```


启用 Windows 自动更新

以下命令将 Windows Update 配置为在每天晚上 10:00 自动下载和安装更新。

```
$configureWindowsUpdateCommand = Send-SSMCommand -InstanceId Instance-ID -DocumentName  
'AWS-ConfigureWindowsUpdate' -Parameters @{'updateLevel'='InstallUpdatesAutomatically';  
'scheduledInstallDay'='Daily'; 'scheduledInstallTime'='22:00'}
```

查看用于启用 Windows 自动更新的命令状态

以下命令使用命令 ID 获取用于启用 Windows 自动更新的命令执行的状态。

```
Get-SSMCommandInvocation -Details $true -CommandId $configureWindowsUpdateCommand.CommandId  
| select -ExpandProperty CommandPlugins
```

禁用 Windows 自动更新

以下命令降低 Windows Update 通知级别，使系统检查更新但不自动更新实例。

```
$configureWindowsUpdateCommand = Send-SSMCommand -InstanceId Instance-ID -DocumentName  
'AWS-ConfigureWindowsUpdate' -Parameters @{'updateLevel'='NeverCheckForUpdates'}
```

查看用于禁用 Windows 自动更新的命令状态

以下命令使用命令 ID 获取用于禁用 Windows 自动更新的命令执行的状态。

```
Get-SSMCommandInvocation -Details $true -CommandId $configureWindowsUpdateCommand.CommandId  
| select -ExpandProperty CommandPlugins
```

使用 AWS-UpdateEC2Config 文档更新 EC2Config

利用 Run Command 和 AWS-EC2ConfigUpdate 文档，您可以更新正在 Windows 实例上运行的 EC2Config 服务。此命令可将 EC2Config 服务更新为最新版本或您指定的版本。

查看说明和可用参数

```
Get-SSMDocumentDescription -Name "AWS-UpdateEC2Config"
```

查看有关参数的更多信息

```
Get-SSMDocumentDescription -Name "AWS-UpdateEC2Config" | select -ExpandProperty Parameters
```

将 EC2Config 更新为最新版本

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName "AWS-UpdateEC2Config"
```

获取具有实例的响应数据的命令信息

此命令从上一个 Send-SSMCommand 返回指定命令的输出：

```
Get-SSMCommandInvocation -CommandId ID -Details $true -InstanceId Instance-ID | select -  
ExpandProperty CommandPlugins
```

将 EC2Config 更新为特定版本

以下命令将 EC2Config 降级到较旧版本：

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName "AWS-UpdateEC2Config" -Parameter @{'version'='3.8.354'; 'allowDowngrade'='true'}
```

使用 Run Command 管理 Windows 更新

Run Command 包含三个文档，可帮助您管理 Amazon EC2 Windows 实例的更新。

- AWS-FindWindowsUpdates - 扫描实例并确定缺少的更新。
- AWS-InstallMissingWindowsUpdates - 在 EC2 实例上安装缺少的更新。
- AWS-InstallSpecificUpdates - 安装特定更新。

Note

有关在使用 Run Command 调用脚本时重启服务器和实例的信息，请参阅[通过脚本重启托管实例 \(p. 187\)](#)。

以下示例演示如何执行指定的 Windows Update 管理任务。

搜索所有缺少的 Windows 更新

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-FindWindowsUpdates' -Parameters @{'UpdateLevel'='All'}
```

安装特定的 Windows 更新

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallSpecificWindowsUpdates' -Parameters @{'KbArticleIds'='123456,KB567890,987654'}
```

安装缺少的重要 Windows 更新

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallMissingWindowsUpdates' -Parameters @{'UpdateLevel'='Important'}
```

安装缺少的 Windows 更新 (带特定排除内容)

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName 'AWS-InstallMissingWindowsUpdates' -Parameters @{'UpdateLevel'='All'; 'ExcludeKbArticleIds'='KB567890,987654'}
```

Systems Manager Run Command 疑难解答

Run Command 在每次命令执行时提供状态详细信息。有关命令状态的更多信息，请参阅[了解命令状态 \(p. 189\)](#)。您也可以使用本主题中的信息来帮助排查 Run Command 问题。

主题

- [我的实例在哪里？ \(p. 202\)](#)
- [获取有关 Windows 实例的状态信息 \(p. 202\)](#)

- [获取有关 Linux 实例的状态信息 \(p. 203\)](#)
- [排除 SSM 代理的故障 \(p. 203\)](#)

我的实例在哪里？

在 Run a command (运行命令) 页面上，在选择要运行的 SSM 文档并在 Targets (目标) 部分中选择 Manually selecting instances (手动选择实例) 后，将显示实例的列表，您可以从中选择要对其运行命令的实例。如果未列出您希望看到的实例，请检查以下要求：

- SSM 代理：确保已在实例上安装最新版本的 SSM 代理。仅 Amazon EC2 Windows Amazon Machine Image (AMI) 和一些 Linux AMI 预配置了 SSM 代理。有关在实例上安装或重新安装 SSM 代理的信息，请参阅 [在 Linux 实例上安装和配置 SSM 代理 \(p. 17\)](#) 或在 [Windows 实例上安装和配置 SSM 代理 \(p. 15\)](#)。
- IAM 实例角色：验证实例是否配置了 AWS Identity and Access Management (IAM) 角色，使得此实例能够与 Systems Manager API 通信。此外，还验证您的用户账户是否具有一个允许您的账户与 Systems Manager API 通信的 IAM 用户信任策略。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。
- 目标操作系统类型：请复查您已选择支持要更新的实例类型的 SSM 文档。大多数 SSM 文档同时支持 Windows 实例和 Linux 实例，但有一些 SSM 文档不支持。例如，如果您选择 SSM 文档 `AWS-InstallPowerShellModule`（此文档仅适用于 Windows 实例），则目标实例列表中将不会显示 Linux 实例。

使用 Health API 检查实例状态

您可以使用 Amazon EC2 Health API 快速确定以下有关 Amazon EC2 实例的信息：

- 一个或多个实例的状态
- 该实例上次发送检测信号值的时间
- SSM 代理的版本
- 操作系统
- EC2Config 服务的版本 (Windows)
- EC2Config 服务的状态 (Windows)

获取有关 Windows 实例的状态信息

使用以下命令可获取有关一个或多个实例的状态详细信息：

```
Get-SSMInstanceInformation -InstanceInformationFilterList
@{Key="InstanceIds";ValueSet="instance-ID","instance-ID"}
```

使用以下不带筛选条件的命令可查看注册到您的账户的当前报告联机状态的所有实例。将 ValueSet="Online" 替换为 "ConnectionLost" 或 "Inactive" 可查看这些状态：

```
Get-SSMInstanceInformation -InstanceInformationFilterList
@{Key="PingStatus";ValueSet="Online"}
```

使用以下命令可查看哪些实例正在运行 EC2Config 服务的最新版本。将 ValueSet="LATEST" 替换为特定版本 (例如，3.0.54 或 3.10) 可查看这些详细信息：

```
Get-SSMInstanceInformation -InstanceInformationFilterList
@{Key="AgentVersion";ValueSet="LATEST"}
```

获取有关 Linux 实例的状态信息

使用以下命令可获取有关一个或多个实例的状态详细信息：

```
aws ssm describe-instance-information --instance-information-filter-list
key=InstanceIds,valueSet=instance-ID
```

使用以下不带筛选条件的命令可查看注册到您的账户的当前报告联机状态的所有实例。将 ValueSet="Online" 替换为 "ConnectionLost" 或 "Inactive" 可查看这些状态：

```
aws ssm describe-instance-information --instance-information-filter-list
key=PingStatus,valueSet=Online
```

使用以下命令可查看哪些实例正在运行 SSM 代理的最新版本。将 ValueSet="LATEST" 替换为特定版本 (例如, 1.0.145 或 1.0) 可查看这些详细信息：

```
aws ssm describe-instance-information --instance-information-filter-list
key=AgentVersion,valueSet=LATEST
```

如果 describe-instance-information API 操作返回的 AgentStatus 为 Online，则可使用 Run Command 来管理实例。如果状态为 "Inactive"，则该实例具有下面的一个或多个问题。

- 未安装 SSM 代理。
- 实例没有出站 Internet 连接。
- 未使用支持实例与 SSM API 通信的 IAM 角色来启动实例，或 IAM 角色的权限对 Run Command 不适用。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

排除 SSM 代理的故障

如果您在使用 Run Command 执行命令时遇到问题，则可能是因为 SSM 代理有问题。使用以下信息可帮助您排查代理问题。

查看代理日志

SSM 代理在下列日志文件中记录信息。这些文件中的信息可帮助您排查问题。

在 Windows 上

- %PROGRAMDATA%\Amazon\SSM\Logs\amazon-ssm-agent.log
- %PROGRAMDATA%\Amazon\SSM\Logs\error.log

Note

seelog 的默认文件名为 seelog.xml.template。如果修改 seelog，必须将文件重命名为 seelog.xml。

在 Linux 上

- /var/log/amazon/ssm/amazon-ssm-agent.log
- /var/log/amazon/ssm/errors.log

在 Linux 上，您可通过更新 seelog.xml 文件来启用延长日志记录。默认情况下，配置文件位于以下位置：/etc/amazon/ssm/seelog.xml。

有关 `cihub/seelog` 配置的更多信息，请转到 [cihub/seelog Wiki](#)。有关 `cihub/seelog` 配置的示例，请转到 [cihub/seelog 示例](#)。

AWS Systems Manager Patch Manager

AWS Systems Manager Patch Manager 能够使用安全相关更新自动执行修补托管实例的过程。对于基于 Linux 的实例，您还可以安装适用于非安全性更新的补丁。可以按操作系统类型对 Amazon EC2 实例或本地服务器和虚拟机 (VM) 的队列进行修补。这包括受支持版本的 Windows、Ubuntu Server、Red Hat Enterprise Linux (RHEL)、SUSE Linux Enterprise Server (SLES)、Amazon Linux 和 Amazon Linux 2。您可以扫描实例以仅查看缺失补丁的报告，也可以扫描并自动安装所有缺失的补丁。

Important

在 Patch Manager 中提供 Windows 或 Linux 补丁之前，AWS 不会测试这些补丁。

Patch Manager 使用补丁基准，该基准包含用于在补丁发行几天内自动批准补丁的规则以及一系列已批准和已拒绝的补丁。您可以通过安排修补作为 Systems Manager Maintenance Window 任务运行来定期安装补丁。您还可以单独安装补丁，或使用 Amazon EC2 标签将补丁安装到大型实例组。

Patch Manager 与 AWS Identity and Access Management (IAM)、AWS CloudTrail 和 Amazon CloudWatch Events 集成，以提供包括事件通知和审核使用情况的能力的安全修补体验。

开始使用 Patch Manager

要开始使用 Patch Manager，请完成下表中描述的任务。

任务	了解更多信息
验证 Systems Manager 先决条件	Systems Manager 先决条件 (p. 6)
了解如何设置和配置修补	使用 Patch Manager (p. 219)
为 Maintenance Window 配置权限 (如果您打算在修补时使用此功能，则需要此操作)。	控制对 Maintenance Window 的访问权限 (p. 247)
创建补丁基准、补丁组和 Maintenance Window，以在测试环境中执行修补	Systems Manager Patch Manager 演练 (p. 225)

主题

- [Patch Manager 支持的操作系统 \(p. 204\)](#)
- [Patch Manager 工作原理 \(p. 205\)](#)
- [介绍如何修补实例的 SSM 文档概述 \(p. 215\)](#)
- [关于 SSM 文档 AWS-RunPatchBaseline \(p. 217\)](#)
- [使用 Patch Manager \(p. 219\)](#)
- [Systems Manager Patch Manager 演练 \(p. 225\)](#)
- [Patch Manager 的 AWS CLI 命令 \(p. 234\)](#)

Patch Manager 支持的操作系统

Patch Manager 功能不支持其他 AWS Systems Manager 功能支持的所有相同的操作系统版本。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

Patch Manager 可用于修补下表中列出的操作系统。

操作系统	详细信息
Linux	<p>仅 64 位系统</p> <ul style="list-style-type: none">• Red Hat Enterprise Linux (RHEL) 7.0 - 7.4• SUSE Linux Enterprise Server (SLES) 12• Amazon Linux 2015.03 - 2018.03• Amazon Linux 2 2-2.0• CentOS 7.1 和更高版本 <p>64 位和 32 位系统</p> <ul style="list-style-type: none">• Red Hat Enterprise Linux (RHEL) 6.5 - 6.9• Ubuntu Server 14.04 LTS 和 16.04 LTS• Amazon Linux 2012.03 - 2017.03• CentOS 6.5 和更高版本 <p>Note</p> <p>从 Amazon Linux AMI 创建的使用代理的实例必须运行 Python requests 模块的当前版本，才能支持 Patch Manager 操作。有关更多信息，请参阅 在使用代理服务器的 Amazon Linux 实例上升级 Python 请求模块 (p. 28)。</p>
Windows	Windows Server 2008 至 Windows Server 2016，包括 R2 版本。在 Microsoft 发布适用于受支持的操作系统的补丁后的几小时内，Patch Manager 将提供所有这些补丁。

有关 Systems Manager 支持的全部操作系统的列表，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

Patch Manager 工作原理

此节介绍一些技术细节，说明 Patch Manager 如何确定安装哪些补丁，以及它如何在每个支持的操作系统上安装这些补丁。对于 Linux 操作系统，它还提供有关在自定义补丁基准中为补丁指定源存储库而非实例上配置的默认源存储库的信息。此节还提供了有关补丁基准规则在不同的 Linux 操作系统发行版上的工作原理的详细信息。

主题

- [如何选择安全性补丁 \(p. 205\)](#)
- [如何指定备用补丁源存储库 \(Linux\) \(p. 208\)](#)
- [如何安装补丁 \(p. 209\)](#)
- [补丁基准规则在基于 Linux 的系统上的工作原理 \(p. 211\)](#)

如何选择安全性补丁

Patch Manager 的主要设计意图在于为实例安装与安全性相关的操作系统更新。默认情况下，Patch Manager 并非安装所有可用的补丁，而只安装一小部分旨在提高安全性的补丁。

Note

在 Patch Manager 支持的所有基于 Linux 的系统上，您可以选择为实例配置的不同源存储库，通常用于安装非安全性更新。有关信息，请参阅 [如何指定备用补丁源存储库 \(Linux\) \(p. 208\)](#)。

本部分的剩余内容解释了 Patch Manager 如何为其他受支持的操作系统选择安全补丁。

Windows

在 Microsoft Windows 操作系统上，Patch Manager 使用 Microsoft 的 cab 文件 `wsusscn2.cab` 作为可用操作系统安全更新的来源。此文件包含有关 Microsoft 发布的与安全相关的更新的信息。Patch Manager 定期从 Microsoft 下载此文件，并使用它更新 Windows 实例可用的补丁集。此文件仅包含 Microsoft 确定与安全有关的更新。在处理文件中的信息时，Patch Manager 还会删除已被后续更新取代的更新。因此，Patch Manager 只显示最新更新，以供您安装。例如，如果 KB4012214 取代了 KB3135456，则 Patch Manager 只将 KB4012214 显示为更新。

要了解有关 `wsusscn2.cab` 文件的更多信息，请参阅 Microsoft 文章 [Using WUA to Scan for Updates Offline](#)。

下载最新版本的 `wsusscn2.cab`：

- [wsusscn2.cab](#)

Amazon Linux 和 Amazon Linux 2

在 Amazon Linux 和 Amazon Linux 2 上，Systems Manager 补丁基准服务使用实例上的预配置存储库。实例上通常有两个预配置存储库 (存储库)：

- 存储库 ID：amzn-main/latest
存储库名称：amzn-main-Base
- 存储库 ID：amzn-updates/latest
存储库名称：amzn-updates-Base

Note

所有更新都是从实例上配置的远程存储库下载的。因此，实例必须能够连接远程存储库，才能执行修补。

Amazon Linux 和 Amazon Linux 2 实例使用 Yum 作为程序包管理器，并且 Yum 使用更新通知的概念。更新通知只是修复特定问题的程序包的集合。Patch Manager 将更新通知中的所有程序包视为安全性程序包。因为没有为单个程序包提供分类或严重性，所以 Patch Manager 为程序包分配它们属于的更新通知的属性。要处理不在更新通知中的程序包，请使用规则中的 `EnableNonSecurity` 标志。

RHEL

在 Red Hat Enterprise Linux 上，Systems Manager 补丁基准服务使用实例上的预配置存储库 (存储库)。实例上通常有三个预配置存储库：

- 存储库 ID：rhui-REGION-client-config-server-7/x86_64
存储库名称：Red Hat Update Infrastructure 2.0 Client Configuration Server 7
- 存储库 ID：rhui-REGION-rhel-server-releases/7Server/x86_64
存储库名称：Red Hat Enterprise Linux Server 7 (RPMs)
- 存储库 ID：rhui-REGION-rhel-server-rh-common/7Server/x86_64
存储库名称：Red Hat Enterprise Linux Server 7 RH Common (RPMs)

Note

所有更新都是从实例上配置的远程存储库下载的。因此，实例必须能够连接远程存储库，才能执行修补。

Red Hat Enterprise Linux 实例使用 Yum 作为程序包管理器，并且 Yum 使用更新通知的概念。更新通知只是修复特定问题的程序包的集合。Patch Manager 将更新通知中的所有程序包视为安全性程序包。因为没有为单个程序包提供分类或严重性，所以 Patch Manager 为程序包分配它们属于的更新通知的属性。要处理不在更新通知中的程序包，请使用规则中的 EnableNonSecurity 标志。

Ubuntu

在 Ubuntu Server 上，Systems Manager 补丁基准服务使用实例上的预配置存储库（存储库）。这些预配置存储库用于提取可用程序包升级的更新列表。在这一点上，Systems Manager 的作用类似于 `sudo apt-get update` 命令。

然后，从 `codename-security` 存储库筛选程序包，其中代号为与 `trusty` 或 `xenial` 类似的内容。例如，在 Ubuntu Server 14 上，Patch Manager 只识别属于 `trusty-security` 一部分的升级。在 Ubuntu Server 16 上，仅标识作为 `xenial-security` 的一部分的升级。

SLES

在 SUSE Linux Enterprise Server (SLES) 实例上，ZYPP 库从以下位置获取可用补丁的列表（程序包的集合）：

- 存储库的列表：`etc/zypp/repos.d/*`
- 程序包信息：`/var/cache/zypp/raw/*`

SLES 实例使用 Zypper 作为程序包管理器，并且 Zypper 使用补丁的概念。补丁只是修复特定问题的程序包的集合。Patch Manager 将补丁中引用的所有程序包都处理为与安全性相关。因为没有为单个程序包提供分类或严重性，所以 Patch Manager 为程序包分配它们属于的补丁的属性。

CentOS

在 CentOS 上，Systems Manager 补丁基准服务使用实例上的预配置存储库（存储库）。下面是 CentOS 6.9 Amazon Machine Image (AMI) 中的一些示例：

- 存储库 ID：`ultra-centos-6.9-base`
存储库名称：`UltraServe CentOS-6.9 - Base`
- 存储库 ID：`ultra-centos-6.9-extras`
存储库名称：`UltraServe CentOS-6.9 - Extras`
- 存储库 ID：`ultra-centos-6.9-updates`
存储库名称：`UltraServe CentOS-6.9 - Updates`
- 存储库 ID：`ultra-centos-6.x-glusterfs`
存储库名称：`UltraServe CentOS-6.x - GlusterFS`
- 存储库 ID：`ultra-centos-6.x-ultrarepo`
存储库名称：`UltraServe CentOS-6.x – UltraServe Repo Packages`

Note

所有更新都是从实例上配置的远程存储库下载的。因此，实例必须能够连接远程存储库，才能执行修补。

CentOS 实例使用 Yum 作为程序包管理器，并且 Yum 使用更新通知的概念。更新通知只是修复特定问题的程序包的集合。Patch Manager 将更新通知中的所有程序包视为安全性程序包。

但是，CentOS 默认存储库不配置更新通知。这意味着，Patch Manager 不检测默认 CentOS 存储库上的程序包。要允许 Patch Manager 处理更新通知中未包含的程序包，您必须启用补丁基准规则中的 `EnableNonSecurity` 标志。

Note

支持 CentOS 更新通知。带有更新通知的存储库发布后即可供下载。

如何指定备用补丁源存储库 (Linux)

当您使用实例上配置的默认存储库执行修补操作时，Patch Manager 会扫描或安装与安全性相关的补丁。这是 Patch Manager 的默认行为。有关 Patch Manager 如何选择和安装安全性补丁的完整信息，请参阅[如何选择安全性补丁 \(p. 205\)](#)。

不过，在 Linux 系统上，您还可以使用 Patch Manager 来安装与安全性无关的补丁，或与实例上配置的默认源存储库不同的源存储库中的补丁。您可以在创建自定义补丁基准时指定备用补丁源存储库。在每个自定义补丁基准中，您可以为多达 20 个版本的受支持的 Linux 操作系统指定补丁源配置。

例如，如果 Ubuntu Server 队列同时包括 Ubuntu Server 14.04 和 Ubuntu Server 16.04 实例，您可以在同一自定义补丁基准中为每个版本指定备用存储库。对于每个版本，您提供名称，指定操作系统版本类型 (产品)，并提供存储库配置。您也可以指定适用于所有版本的受支持的操作系统的一个备用源存储库。

有关使用此选项的示例场景的列表，请参阅本主题后面的[备用补丁源存储库的示例用法 \(p. 208\)](#)。

有关默认和自定义补丁基准的信息，请参阅[验证默认补丁基准或创建自定义补丁基准 \(p. 220\)](#)。

Note

运行指定实例上的备用补丁存储库的自定义补丁基准不会更改为实例配置的默认存储库。

使用控制台

要在 AWS Systems Manager 控制台中指定备用补丁源存储库，请使用创建补丁基准页面上的补丁来源部分。有关使用补丁来源选项的信息，请参阅[演练：修补服务器环境 \(控制台\) \(p. 225\)](#) 主题的[创建默认补丁基准 \(p. 225\)](#)部分。

使用其他工具创建补丁基准

在创建补丁基准时，将 `sources` 选项与其他工具结合使用。

- AWS CLI : [create-patch-baseline](#)
- Systems Manager API : [API_CreatePatchBaseline](#)
- Systems Manager AWS Tools for Windows PowerShell : [New-SSMPatchBaseline](#)

有关将 `--sources` 选项与 CLI 结合使用的示例，请参阅[创建对不同操作系统版本使用自定义存储库的补丁基准 \(p. 235\)](#)。

备用补丁源存储库的示例用法

示例 1 – Ubuntu Server 的非安全性更新

您已使用 Patch Manager 来通过 AWS 提供的默认补丁基准 `AWS-UbuntuDefaultPatchBaseline` 在 Ubuntu Server 实例队列上安装安全性补丁。您可以创建基于此默认补丁基准的新补丁基准，但在批准规则中指定您也希望安装属于默认分配的一部分的与安全性无关的更新。当对您的实例运行此补丁基准时，将应用针对安全性和非安全性问题的补丁。您还可以选择在为基准指定的补丁异常中批准非安全性补丁。

示例 2 – Ubuntu Server 的个人程序包存档 (PPA)

您的 Ubuntu Server 实例运行通过 [Ubuntu 的个人程序包存档 \(PPA\)](#) 分配的软件。在这种情况下，创建将您在实例上配置的 PPA 存储库指定为修补操作的源存储库的补丁基准。然后使用 Run Command 在实例上运行补丁基准文档。

示例 3 – Amazon Linux 上的企业内部应用程序

您需要在 Amazon Linux 实例上运行行业监管合规性所需的一些应用程序。您可以为实例上的这些应用程序配置存储库，使用 YUM 对这些应用程序进行初始安装，然后更新或创建新的补丁基准以包括此新企业存储库。在此之后，您可以在实例上使用 Run Command 运行 AWS-RunPatchBaseline 文档，通过 Scan 选项来查看企业程序包是否列在已安装程序包中以及是否最新。如果它不是最新的，可以使用 Install 选项再次运行该文档来更新应用程序。

如何安装补丁

Patch Manager 使用操作系统类型的相应内置机制在实例上安装更新。例如，在 Windows 上，使用 Windows Update API；在 Amazon Linux 上，则使用 yum 程序包管理器。

本部分的剩余内容解释了 Patch Manager 如何在操作系统上安装补丁。

Windows

在 Windows 实例上执行修补操作时，实例从 Systems Manager 请求相应补丁基准的快照。此快照包含已批准部署的补丁基准中可用的所有更新的列表。将更新列表发送到 Windows Update API，Windows Update API 确定哪些更新适用于实例并根据需要进行安装。如果安装了任意更新，则根据完成所有必要修补所需的次数重启实例。可在 Run Command 请求输出中找到修补操作摘要。其他日志可在实例上的 %PROGRAMDATA%\Amazon\PatchBaselineOperations\Logs 文件夹中找到。

因为使用 Windows Update API 下载和安装补丁，所以操作遵循 Windows Update 的所有组策略设置。使用 Patch Manager 时不必进行任何组策略设置，但已定义的任何设置都有效，例如，将实例定向到 WSUS 服务器。

Note

默认情况下，Windows 从 Microsoft 的 Windows Update 站点下载所有补丁，这是因为 Patch Manager 使用 Windows Update API 驱动补丁的下载和安装。因此，实例必须能够访问 Microsoft Windows Update 站点，否则修补将失败。或者，您也可以将 WSUS 服务器配置为补丁存储库，然后使用组策略将实例配置为定位到此 WSUS 服务器。

Amazon Linux 和 Amazon Linux 2

在 Amazon Linux 和 Amazon Linux 2 实例上，补丁安装工作流程如下：

1. 依照补丁基准中的规定应用 [GlobalFilters](#)，只保留符合资格的程序包做进一步处理。
2. 依照补丁基准中的规定应用 [ApprovalRules](#)。每条批准规则可以将一个程序包定义为已批准。
3. 依照补丁基准中的规定应用 [ApprovedPatches](#)。已批准的补丁即使被 [GlobalFilters](#) 丢弃，也会批准更新；如果 [ApprovalRules](#) 中未指定任何批准规则，则给予批准。
4. 依照补丁基准中的规定应用 [RejectedPatches](#)。从已批准的补丁列表中删除已拒绝的补丁，并且不会应用已拒绝的补丁。
5. 如果批准了补丁的多个版本，则应用最新版本。
6. 对已批准的补丁应用 YUM 更新 API。
7. 如果安装了任意更新，则重启实例。

Note

此工作流程的等效 yum 命令为：

```
sudo yum update-minimal --security --bugfix
```

RHEL

在 Red Hat Enterprise Linux 实例上，补丁安装工作流程如下：

1. 依照补丁基准中的规定应用 [GlobalFilters](#)，只保留符合资格的程序包做进一步处理。
2. 依照补丁基准中的规定应用 [ApprovalRules](#)。每条批准规则可以将一个程序包定义为已批准。
3. 依照补丁基准中的规定应用 [ApprovedPatches](#)。已批准的补丁即使被 [GlobalFilters](#) 丢弃，也会批准更新；如果 [ApprovalRules](#) 中未指定任何批准规则，则给予批准。
4. 依照补丁基准中的规定应用 [RejectedPatches](#)。从已批准的补丁列表中删除已拒绝的补丁，并且不会应用已拒绝的补丁。
5. 如果批准了补丁的多个版本，则应用最新版本。
6. 对已批准的补丁应用 YUM 更新 API。
7. 如果安装了任意更新，则重启实例。

Note

此工作流程的等效 yum 命令为：

```
sudo yum update-minimal --security --bugfix
```

Ubuntu

在 Ubuntu Server 实例上，补丁安装工作流程如下：

1. 依照补丁基准中的规定应用 [GlobalFilters](#)，只保留符合资格的程序包做进一步处理。
2. 依照补丁基准中的规定应用 [ApprovalRules](#)。每条批准规则可以将一个程序包定义为已批准。此外，还应用了一条隐式规则，以便只选择在安全存储库中有升级的程序包。对于每个程序包，程序包的候选版本 (通常为最新版本) 必须包含在安全存储库中。
3. 依照补丁基准中的规定应用 [ApprovedPatches](#)。已批准的补丁即使被 [GlobalFilters](#) 丢弃，也会批准更新；如果 [ApprovalRules](#) 中未指定任何批准规则，则给予批准。
4. 依照补丁基准中的规定应用 [RejectedPatches](#)。从已批准的补丁列表中删除已拒绝的补丁，并且不会应用已拒绝的补丁。
5. 使用 APT 库升级程序包。
6. 如果安装了任意更新，则重启实例。

SLES

在 SUSE Linux Enterprise Server (SLES) 实例上，补丁安装工作流程如下：

1. 依照补丁基准中的规定应用 [GlobalFilters](#)，只保留符合资格的程序包做进一步处理。
2. 依照补丁基准中的规定应用 [ApprovalRules](#)。每条批准规则可以将一个程序包定义为已批准。
3. 依照补丁基准中的规定应用 [ApprovedPatches](#)。已批准的补丁即使被 [GlobalFilters](#) 丢弃，也会批准更新；如果 [ApprovalRules](#) 中未指定任何批准规则，则给予批准。
4. 依照补丁基准中的规定应用 [RejectedPatches](#)。从已批准的补丁列表中删除已拒绝的补丁，并且不会应用已拒绝的补丁。
5. 如果批准了补丁的多个版本，则应用最新版本。
6. 对已批准的补丁应用 Zypper 更新 API。
7. 如果安装了任意更新，则重启实例。

CentOS

在 CentOS 实例上，补丁安装工作流程如下：

1. 依照补丁基准中的规定应用 [GlobalFilters](#)，只保留符合资格的程序包做进一步处理。

2. 依照补丁基准中的规定应用 [ApprovalRules](#)。每条批准规则可以将一个程序包定义为已批准。
3. 依照补丁基准中的规定应用 [ApprovedPatches](#)。已批准的补丁即使被 [GlobalFilters](#) 丢弃，也会批准更新；如果 [ApprovalRules](#) 中未指定任何批准规则，则给予批准。
4. 依照补丁基准中的规定应用 [RejectedPatches](#)。从已批准的补丁列表中删除已拒绝的补丁，并且不会应用已拒绝的补丁。
5. 如果批准了补丁的多个版本，则应用最新版本。
6. 对已批准的补丁应用 YUM 更新 API。
7. 如果安装了任意更新，则重启实例。

补丁基准规则在基于 Linux 的系统上的工作原理

对于 Linux 发行版，补丁基准中规则的工作方式因发行版类型的不同而有所差异。与 Windows 实例上的补丁更新不同，规则将在每个实例上进行评估，以考虑此实例上配置的存储库。Patch Manager 使用本机程序包管理器驱动补丁基准批准的补丁的安装。

主题

- [补丁基准规则在 Amazon Linux 和 Amazon Linux 2 上的工作原理 \(p. 211\)](#)
- [补丁基准规则在 RHEL 上的工作原理 \(p. 212\)](#)
- [补丁基准规则在 Ubuntu Server 上的工作原理 \(p. 213\)](#)
- [补丁基准规则在 SUSE Linux Enterprise Server 上的工作原理 \(p. 214\)](#)

补丁基准规则在 Amazon Linux 和 Amazon Linux 2 上的工作原理

在 Amazon Linux 和 Amazon Linux 2 上，补丁选择过程如下：

1. 在实例上，YUM 库访问每个配置的存储库的 `updateinfo.xml` 文件。

Note

如果存储库不是由 Amazon 管理的，可能不存在 `updateinfo.xml` 文件。如果找不到 `updateinfo.xml`，则不会应用任何补丁。

2. `updateinfo.xml` 中的每个更新通知都包含几个属性，它们表示通知中的程序包的属性，如下表所述。

更新通知属性

属性	说明
type	<p>对应于补丁基准的 PatchFilter 数据类型中 <code>Classification</code> 键属性的值。表示更新通知中包含的程序包的类型。</p> <p>可能的值：</p> <ul style="list-style-type: none">• 安全性• Bugfix• Enhancement• 推荐使用• Newpackage
severity	<p>对应于补丁基准的 PatchFilter 数据类型中 <code>Severity</code> 键属性的值。表示更新通知中包含的程序包的严重性。通常只适用于安全性更新通知。</p> <p>可能的值：</p>

属性	说明
	<ul style="list-style-type: none">• 重大• 重要提示• 中等• Low
update_id	表示建议 ID，例如 ALAS-2017-867。补丁基准中的 ApprovedPatches 或 RejectedPatches 属性可能会使用建议 ID。
重点	包含有关更新通知的其他信息，例如 CVE ID (格式：CVE-2017-1234567)。补丁基准中的 ApprovedPatches 或 RejectedPatches 属性可以使用 CVE ID。
updated	对应于补丁基准中的 ApproveAfterDays 。表示更新通知中包含的程序包的发布日期 (更新的日期)。将当前时间戳与此属性的值比较并配合 ApproveAfterDays 可用来确定补丁是否已获得部署批准。

Note

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式 \(p. 222\)](#)。

3. 实例的产品由 SSM 代理确定。此属性对应于补丁基准的 [PatchFilter](#) 数据类型中 Product 键属性的值。
4. 对于 updateinfo.xml 中的每个更新通知，补丁基准用作筛选器，只允许更新包含符合条件的程序包。如果应用补丁基准定义后有多个程序包适用，则使用最新版本。

补丁基准规则在 RHEL 上的工作原理

在 Red Hat Enterprise Linux 上，补丁选择过程如下：

1. 在实例上，YUM 库访问每个配置的存储库的 updateinfo.xml 文件。

Note

如果存储库不是由 Red Hat 管理的，可能不存在 updateinfo.xml 文件。如果找不到 updateinfo.xml，则不会应用任何补丁。

2. updateinfo.xml 中的每个更新通知都包含几个属性，它们表示通知中的程序包的属性，如下表所述。

更新通知属性

属性	说明
type	<p>对应于补丁基准的 PatchFilter 数据类型中 Classification 键属性的值。表示更新通知中包含的程序包的类型。</p> <p>可能的值：</p> <ul style="list-style-type: none">• 安全性• Bugfix• Enhancement• 推荐使用

属性	说明
	<ul style="list-style-type: none"> Newpackage
severity	<p>对应于补丁基准的 PatchFilter 数据类型中 Severity 键属性的值。表示更新通知中包含的程序包的严重性。通常只适用于安全性 更新通知。</p> <p>可能的值：</p> <ul style="list-style-type: none"> 重大 重要提示 适中 Low None (如果更新通知中未指定严重性，或者为空字符串。)
update_id	表示建议 ID，例如 RHSA-2017:0864。补丁基准中的 ApprovedPatches 或 RejectedPatches 属性可能会使用建议 ID。
重点	包含有关更新通知的其他信息，例如 CVE ID (格式：CVE-2017-1000371) 或 Bugzilla ID (格式：1463241)。补丁基准中的 ApprovedPatches 或 RejectedPatches 属性可以使用 CVE ID 和 Bugzilla ID。
updated	对应于补丁基准中的 ApproveAfterDays 。表示更新通知中包含的程序包的发布日期 (更新的日期)。将当前时间戳与此属性的值比较并配合 ApproveAfterDays 用来确定补丁是否已获得部署批准。

Note

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式 \(p. 222\)](#)。

- 实例的产品由 SSM 代理确定。此属性对应于补丁基准的 [PatchFilter](#) 数据类型中 Product 键属性的值。
- 对于 updateinfo.xml 中的每个更新通知，补丁基准用作筛选器，只允许更新包含符合条件的程序包。如果应用补丁基准定义后有多个程序包适用，则使用最新版本。

补丁基准规则在 Ubuntu Server 上的工作原理

在 Ubuntu Server 上，补丁基准服务提供对 Priority 和 Section 字段的筛选。这些字段通常存在于所有 Ubuntu Server 程序包中。为确定补丁基准是否选择了某个补丁，Patch Manager 执行以下操作：

- 在 Ubuntu 系统上，运行等效于 `sudo apt-get update` 的程序刷新可用程序包列表。不配置存储库，从 sources 列表中配置的存储库提取数据。
- 接下来，应用 [GlobalFilters](#)、[ApprovalRules](#)、[ApprovedPatches](#) 和 [RejectedPatches](#) 列表。只选择发行版安全存储库 (存档) 中存在候选版本的程序包。对于 Ubuntu Server 14，此存储库为 `trusty-security`。对于 Ubuntu Server 16，它是 `xenial-security`。

Note

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式 \(p. 222\)](#)。

要查看 Priority 和 Section 字段的内容，运行以下 aptitude 命令：

Note

您需要先在 Ubuntu Server 16 系统上安装 Aptitude。

```
aptitude search -F '%p %P %S %t %V#' '~U'
```

在对此命令的响应中，按以下格式报告所有可升级程序包：

```
name, priority, section, archive, candidate version
```

对于 Ubuntu Server，将程序包分类到不同合规性状态的规则如下：

- 已安装：通过了补丁基准筛选、trusty-security (Ubuntu Server 14) 或 xenial-security (Ubuntu Server 16) 中存在候选版本并且不可升级的程序包。
- 缺少：通过了基准筛选、trusty-security (Ubuntu Server 14) 或 xenial-security (Ubuntu Server 16) 中存在候选版本并且可升级的程序包。
- 已安装其他程序包：未通过基准筛选、trusty-security (Ubuntu Server 14) 或 xenial-security (Ubuntu Server 16) 中存在候选版本并且不可升级的程序包。此类程序包的合规性级别设置为 UNSPECIFIED。
- NotApplicable：ApprovedPatches 中包含但系统上未安装的程序包。
- Failed：修补操作期间安装失败的程序包。

Note

有关适用于所有操作系统的补丁合规性状态值的一般信息，请参阅[关于补丁合规性 \(p. 95\)](#)。

补丁基准规则在 SUSE Linux Enterprise Server 上的工作原理

在 SLES 上，每个补丁包括以下表示补丁中的程序包的属性的属性：

- 类别：对应于补丁基准的 PatchFilter 数据类型中的分类键属性的值。表示更新通知中包含的补丁的类型。可用选项包括：
 - ###
 - ####
 - ##
 - ##
 - ##
 - Yast
- 严重性：对应于补丁基准的 PatchFilter 数据类型中严重性键属性的值。表示补丁的严重性。可用选项包括：
 - #
 - #
 - ##
 - ####
 - ##

实例的产品由 SSM 代理确定。此属性对应于补丁基准的 PatchFilter 数据类型中 Product 键属性的值。

对于每个补丁，补丁基准用作筛选器，只允许更新包含符合条件的程序包。如果应用补丁基准定义后有多多个程序包适用，则使用最新版本。

Note

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式](#) (p. 222)。

介绍如何修补实例的 SSM 文档概述

本主题介绍了目前可用的七个 SSM 文档，有助于您使托管实例始终获得最新的安全相关更新补丁。

目前，我们建议在修补操作中仅使用这些文档中的三个。结合使用这三个 SSM 文档，您即可获得使用 AWS Systems Manager 进行修补的全部选项。其中有两个文档是在四个早期 SSM 文档之后发布的，替换了这四个早期文档，对功能进行了扩展和整合。

这三个推荐的 SSM 文档是：

- AWS-ConfigureWindowsUpdate
- AWS-InstallWindowsUpdates
- AWS-RunPatchBaseline

四个早期 SSM 文档现在仍可使用，但将来可能弃用，它们是：

- AWS-ApplyPatchBaseline
- AWS-FindWindowsUpdates
- AWS-InstallMissingWindowsUpdates
- AWS-InstallSpecificWindowsUpdates

请参考以下各节，了解有关如何在修补操作中使用这些 SSM 文档的更多信息。

主题

- [适用于修补实例的推荐 SSM 文档](#) (p. 215)
- [适用于修补实例的早期 SSM 文档](#) (p. 216)

适用于修补实例的推荐 SSM 文档

建议在进行托管实例的修补操作时使用以下三个 SSM 文档。

推荐的 SSM 文档

- [AWS-ConfigureWindowsUpdate](#) (p. 215)
- [AWS-InstallWindowsUpdates](#) (p. 216)
- [AWS-RunPatchBaseline](#) (p. 216)

AWS-ConfigureWindowsUpdate

支持配置基本的 Windows 更新功能，并使用它们来自动安装更新 (或禁用自动更新)。在所有 AWS 区域中可用。

此 SSM 文档将提示 Windows 更新下载并安装指定的更新，并根据需要重启实例。请将此文档与 状态管理器 结合使用，确保 Windows 更新保持其配置。您也可以使用 Run Command 手动运行它，以更改 Windows 更新配置。

本文档中的可用参数支持指定要安装的更新类别 (或是否禁用自动更新), 还支持指定在星期几的什么时间运行修补操作。如果您不需要对 Windows 更新进行严格控制, 也不需要收集合规性信息, 则此 SSM 文档最为有用。

替换早期 SSM 文档 :

- 无

AWS-InstallWindowsUpdates

在 Windows 实例上安装更新。在所有 AWS 区域中可用。

如果您希望安装指定更新 (使用 `Include Kbs` 参数), 或希望安装特定类别或分类的补丁, 但不需要补丁合规性信息, 则此 SSM 文档可提供基本修补功能。

替换早期 SSM 文档 :

- AWS-FindWindowsUpdates
- AWS-InstallMissingWindowsUpdates
- AWS-InstallSpecificWindowsUpdates

这三个早期文档执行不同的功能, 但您可以使用较新的 SSM 文档 `AWS-InstallWindowsUpdates` 的不同参数设置得到同样的结果。这些参数设置在 [适用于修补实例的早期 SSM 文档 \(p. 216\)](#) 中进行了介绍。

AWS-RunPatchBaseline

在实例上安装补丁, 或扫描实例以确定是否缺少任何符合条件的补丁。在所有 AWS 区域中可用。

可通过 `AWS-RunPatchBaseline` 使用补丁基准控制对补丁的审批。报告补丁合规性信息, 您可以使用 Systems Manager 合规性工具进行查看。您可使用这些工具深入了解您的实例的补丁合规性状态, 例如哪些实例缺少补丁, 以及缺少哪些补丁。对于 Linux 操作系统, 为来自实例上配置的默认源存储库和来自您在自定义补丁基准中指定的任何备用源存储库的补丁提供合规性信息。有关备用源存储库的更多信息, 请参阅 [如何指定备用补丁源存储库 \(Linux\) \(p. 208\)](#)。有关 Systems Manager 合规性工具的更多信息, 请参阅 [AWS Systems Manager 配置合规性 \(p. 92\)](#)。

替换早期文档 :

- AWS-ApplyPatchBaseline

早期文档 `AWS-ApplyPatchBaseline` 仅适用于 Windows 实例。较新的 `AWS-RunPatchBaseline` 对 Windows 和 Linux 系统提供了同等支持。要使用 `AWS-RunPatchBaseline` 文档, 需要 SSM 代理版本 2.0.834.0 或更高版本。

有关 `AWS-RunPatchBaseline` SSM 文档的更多信息, 请参阅 [关于 SSM 文档 AWS-RunPatchBaseline \(p. 217\)](#)。

适用于修补实例的早期 SSM 文档

在进行修补操作时仍可使用以下四个 SSM 文档。但是, 将来它们可能会被弃用, 因此不建议使用。请使用 [适用于修补实例的推荐 SSM 文档 \(p. 215\)](#) 中介绍的文档。

早期 SSM 文档

- [AWS-ApplyPatchBaseline \(p. 217\)](#)
- [AWS-FindWindowsUpdates \(p. 217\)](#)
- [AWS-InstallMissingWindowsUpdates \(p. 217\)](#)
- [AWS-InstallSpecificWindowsUpdates \(p. 217\)](#)

AWS-ApplyPatchBaseline

仅支持 Windows 实例，但在其替换文档 AWS-RunPatchBaseline 中也可找到一组相同的参数。在 2017 年 8 月之后推出的 AWS 区域中不提供。

Note

此 SSM 文档的替换文档 AWS-RunPatchBaseline 需要版本 2.0.834.0 或更高版本的 SSM 代理。可以使用 AWS-UpdateSSMAgent 文档将您的实例更新为代理的最新版本。

AWS-FindWindowsUpdates

被 AWS-InstallWindowsUpdates 替换，新文档可执行所有相同的操作。在 2017 年 4 月之后推出的 AWS 区域中不提供。

要得到与这个早期 SSM 文档相同的结果，请使用推荐的替换文档 AWS-InstallWindowsUpdates 的以下参数配置：

- Action = ##
- Allow Reboot = #

AWS-InstallMissingWindowsUpdates

被 AWS-InstallWindowsUpdates 替换，新文档可执行所有相同的操作。在 2017 年 4 月之后推出的所有 AWS 区域中均不提供。

要得到与这个早期 SSM 文档相同的结果，请使用推荐的替换文档 AWS-InstallWindowsUpdates 的以下参数配置：

- Action = ##
- Allow Reboot = True

AWS-InstallSpecificWindowsUpdates

被 AWS-InstallWindowsUpdates 替换，新文档可执行所有相同的操作。在 2017 年 4 月之后推出的所有 AWS 区域中均不提供。

要得到与这个早期 SSM 文档相同的结果，请使用推荐的替换文档 AWS-InstallWindowsUpdates 的以下参数配置：

- Action = ##
- Allow Reboot = True
- Include Kbs = KB #####

关于 SSM 文档 AWS-RunPatchBaseline

AWS Systems Manager 支持 Patch Manager 的 SSM 文档 AWS-RunPatchBaseline，此文档对实例执行安全修补操作。此文档同时支持 Linux 和 Windows 实例，因此可在由 Systems Manager 管理的任何类型的实例上可靠运行。此文档将为每个平台执行适当的操作。

Note

Patch Manager 还支持旧版 SSM 文档 AWS-ApplyPatchBaseline。但是，此文档只支持在 Windows 实例上进行修补。建议使用 AWS-RunPatchBaseline，因为它同时支持在 Linux 和 Windows 实例上进行修补。要使用 AWS-RunPatchBaseline 文档，需要 SSM 代理版本 2.0.834.0 或更高版本。

在 Windows 系统上：

在 Windows 实例上，AWS-RunPatchBaseline 文档下载并调用 PowerShell 模块，然后由 PowerShell 模块下载适用于该实例的补丁基准的快照。将此补丁基准快照传递给 Windows Update API，Windows Update API 根据需要控制已批准的补丁的下载和安装。

在 Linux 系统上：

在 Linux 实例上，AWS-RunPatchBaseline 文档调用 Python 模块，然后由 Python 模块下载适用于该实例的补丁基准的快照。此补丁基准快照使用已定义的规则及已批准补丁和已阻止补丁的列表驱动每种实例类型相应的程序包管理器：

- Amazon Linux、Amazon Linux 2 和 RHEL 实例使用 YUM。对于 YUM 操作，Patch Manager 需要 Python 2.6 或更高版本。
- Ubuntu Server 实例使用 APT。对于 APT 操作，Patch Manager 需要 Python 3。
- SUSE Linux Enterprise Server 实例使用 Zypper。对于 Zypper 操作，Patch Manager 需要 Python 2.6 或更高版本。

安装完所有已批准和适用的更新后，根据需要执行重启，然后在实例上生成补丁合规性信息，并向 Patch Manager 报告。有关查看补丁合规性数据的信息，请参阅[关于补丁合规性 \(p. 95\)](#)。

AWS-RunPatchBaseline 参数

AWS-RunPatchBaseline 支持两个参数。Operation 参数是必需的。从技术上讲，Snapshot-ID 是可选的，但建议在 Maintenance Window 之外运行 AWS-RunPatchBaseline 时为其提供一个自定义值；而在 Maintenance Window 操作中运行此文档时，让 Patch Manager 自动提供此值。

参数

- 参数名称: [Operation \(p. 218\)](#)
- 参数名称: [Snapshot ID \(p. 218\)](#)

参数名称: Operation

用法：必需。

选项：Scan | Install。

Scan

选择 Scan 选项时，AWS-RunPatchBaseline 确定实例的补丁合规性状态，并向 Patch Manager 报告此信息。Scan 不提示要安装的更新或要重启的实例。相反，此操作会标识缺少哪些已批准并且适用于此实例的更新。

安装

选择 Install 选项时，AWS-RunPatchBaseline 尝试安装实例中缺少的已批准并且适用的更新。在 Install 操作中生成的补丁合规性信息不会列出任何缺少的更新。但是，如果更新的安装因任何原因失败，它会报告处于失败状态的更新。只要在实例上安装了更新，就一定会重启实例，以确保更新正常安装和激活。

参数名称: Snapshot ID

用法：可选。

Snapshot ID 是 Patch Manager 使用的唯一 ID (GUID)，用于确保在单一操作中修补的一组实例都具有完全相同的一组已批准补丁。尽管它定义为可选参数，根据是否在 Maintenance Window 中运行 AWS-RunPatchBaseline，我们还是提供了不同的最佳实践建议，如下表所述。

AWS-RunPatchBaseline 最佳实践

Mode	最佳实践	详细信息
在 Maintenance Window 中运行 AWS-RunPatchBaseline	不要提供快照 ID。Patch Manager 将为您提供。	<p>如果使用 Maintenance Window 运行 AWS-RunPatchBaseline，则不应提供自己生成的快照 ID。这种情况下，Systems Manager 基于 Maintenance Window 执行 ID 提供 GUID 值。这可确保在 Maintenance Window 中为 AWS-RunPatchBaseline 的所有调用使用正确的 ID。</p> <p>如果在这种情况下您指定值，请注意，补丁基准的快照最多保留 24 小时。之后，即使您在快照到期后指定相同的 ID，也将生成新的快照。</p>
在 Maintenance Window 之外运行 AWS-RunPatchBaseline	为快照 ID 生成和指定自定义 GUID 值。 ¹	<p>如果不使用 Maintenance Window 运行 AWS-RunPatchBaseline，建议为每个补丁基准生成并指定一个唯一的快照 ID，特别是在同一操作中在多个实例上多次运行 AWS-RunPatchBaseline 文档时。如果在这种情况下您不指定 ID，Systems Manager 会为命令发送到的每个实例生成不同的快照 ID。这会导致在实例间指定不同的补丁集。</p> <p>例如，假设您通过 Run Command 直接运行 AWS-RunPatchBaseline 文档，目标为包含 50 个实例的实例组。指定自定义快照 ID 将生成单一基准快照 (用于评估和修补所有实例)，从而确保所有实例最终处于一致状态。</p>

¹ 您可以使用任何能够生成 GUID 的工具为快照 ID 参数生成值。例如，在 PowerShell 中，可以使用 New-Guid cmdlet 生成格式为 12345699-9405-4f69-bc5e-9315aEXAMPLE 的 GUID。

使用 Patch Manager

要使用 Patch Manager，请完成以下任务。此部分更详细地说明了这些任务。

1. 验证默认补丁基准是否满足您的需求，或创建为您的实例定义一组标准补丁的补丁基准。
2. 使用 Amazon EC2 标签将实例组织到补丁组中 (可选，但不建议)。
3. 使用定义要修补的实例以及修补这些实例的时间的 Maintenance Window 来计划修补。
4. 监视修补以验证合规性和调查故障。

主题

- [验证默认补丁基准或创建自定义补丁基准 \(p. 220\)](#)

- [将实例组织到补丁组中 \(p. 224\)](#)
- [使用 Maintenance Window 计划补丁更新 \(p. 224\)](#)

相关内容

- 要查看如何创建补丁基准、补丁组和Maintenance Window的示例，请参阅 [Systems Manager Patch Manager 演练 \(p. 225\)](#)。
- 有关Maintenance Window的更多信息，请参阅 [AWS Systems Manager Maintenance Window \(p. 246\)](#)。
- 有关监视补丁合规性的信息，请参阅[关于补丁合规性 \(p. 95\)](#)。

验证默认补丁基准或创建自定义补丁基准

补丁基准定义批准在您的实例上安装的补丁。您可以逐个指定批准或拒绝的补丁。也可以创建自动批准规则，指定应自动批准的某些更新类型 (例如重要更新)。拒绝补丁列表将覆盖这些规则和批准列表。

要使用一系列已批准的补丁来安装特定程序包，需要先删除所有自动批准规则。如果您将补丁显式标识为 rejected，即使它匹配自动批准规则中的所有条件，也不会被批准或安装。此外，即使补丁被批准用于某个实例，只有它适用于此实例上的软件时，才会安装此补丁。

主题

- [预定义基准与自定义基准 \(p. 220\)](#)
- [Windows 和 Linux 修补之间的重要区别 \(p. 222\)](#)
- [已批准补丁和已拒绝补丁列表的程序包名称格式 \(p. 222\)](#)

预定义基准与自定义基准

Patch Manager 为 Patch Manager 支持的每个操作系统提供预定义补丁基准。您可以按照这些基准的当前配置使用这些基准 (您不能对其进行自定义)，或者，如果您想更好地控制批准或拒绝将哪些补丁用于您的环境，则可创建自己的补丁基准。

主题

- [预定义基准 \(p. 220\)](#)
- [自定义基准 \(p. 221\)](#)

预定义基准

下表介绍了 Patch Manager 提供的预定义补丁基准：

Name	支持的产品	详细信息
AWS-DefaultPatchBaseline	Windows (Windows Server 2008 – 2016)	在发布 7 天后批准分类为“重要更新”或“安全更新”且 MSRC 严重性为“重大”或“重要”的所有操作系统补丁。
AWS-AmazonLinuxDefaultPatchBaseline	Amazon Linux (2012.03 – 2018.03) 和 Amazon Linux 2 2-2.0	在发布 7 天后批准分类为“安全性”且严重性为“重大”或“重要”的所有操作系统补丁。在发布 7 天后还批准分类为“缺陷修正”的所有补丁

Name	支持的产品	详细信息
AWS-UbuntuDefaultPatchBaseline	Ubuntu Server (14.04/16.04)	立即批准优先级为“必需”或“重要”的与安全性相关的所有操作系统补丁。由于存储库中未提供可靠的发布日期，因此批准之前无需等待。
AWS-RedHatDefaultPatchBaseline	Redhat Enterprise Linux Red Hat Enterprise Linux (6.5、6.6、6.7、6.8、6.9、7.0、7.1)	在发布 7 天后批准分类为“安全性”且严重性为“重大”或“重要”的所有操作系统补丁。在发布 7 天后还批准分类为“缺陷修正”的所有补丁。
AWS-SuseDefaultPatchBaseline	SUSE Linux Enterprise Server 12	在发布 7 天后批准分类为“安全性”且严重性为“重大”或“重要”的所有操作系统补丁。
AWS-CentOSDefaultPatchBaseline	CentOS 6.5 和更高版本	在所有更新可用后 7 天批准这些更新 (包括安全性之外的更新)。

自定义基准

如果您创建自己的补丁基准，则可使用以下类别选择自动批准哪些补丁。

- 操作系统：Windows、Amazon Linux、Ubuntu Server 等
- 产品名称：例如，RHEL 6.5、Amazon Linux 2014.09、Windows Server 2012、Windows Server 2012 R2 等
- 分类：例如，关键更新、安全更新等
- 严重性：例如，关键、重要等

对于您创建的每个自动批准规则，可指定自动批准延迟。此延迟是发布补丁后到自动批准补丁用于修补前等待的天数。例如，如果您使用关键更新分类创建一条规则并将其自动批准延迟配置为 7 天，则将在 1 月 14 日自动批准 1 月 7 日发布的新关键补丁。

Note

如果 Linux 存储库不提供程序包的发布日期信息，Systems Manager 会将自动批准延迟的值视为零。

您还可以指定合规性严重性级别。如果经批准的补丁报告为缺失，则 Compliance Level 是违反合规性的严重性。

通过使用多个带不同的自动批准延迟的补丁基准，您可以不同的速率将补丁部署到不同的实例。例如，您可以为开发环境和生产环境创建单独的补丁基准和自动批准延迟。这使您能够先在开发环境中测试补丁，然后再在生产环境中部署这些补丁。

创建补丁基准时，请牢记以下信息：

- Patch Manager 为每个受支持的操作系统提供默认补丁基准。您也可以创建自己的补丁基准并将其指定为相应操作系统的默认补丁基准。
- 对于本地或非 Amazon EC2 实例，Patch Manager 将尝试使用您的自定义默认补丁基准。如果不存在自定义默认补丁基准，系统将使用相应操作系统的预定义补丁基准。
- 如果某个补丁在相同的补丁基准中同时被列为已批准和已拒绝，则该补丁将被拒绝。
- 一个实例只能定义一个补丁基准。

- 可以添加到补丁基准的已批准补丁和已拒绝补丁列表中的程序包名称格式取决于您正在修补的操作系统类型。

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式](#) (p. 222)。

要查看如何使用 Systems Manager 控制台或 AWS CLI 创建补丁基准的示例，请参阅 [Systems Manager Patch Manager 演练](#) (p. 225)。

Windows 和 Linux 修补之间的重要区别

下表介绍了 Windows 和 Linux 修补之间的重要区别。

Note

要修补 Linux 实例，您的实例必须运行 SSM 代理 2.0.834.0 版或更高版本。有关更新此代理的信息，请参阅 [从控制台运行命令](#) (p. 182) 中标题为示例：更新 SSM 代理的部分。AWS-ApplyPatchBaseline SSM 文档不支持 Linux 实例。如需对 Windows 和 Linux 实例应用补丁基准，推荐的 SSM 文档是 AWS-RunPatchBaseline。有关更多信息，请参阅 [介绍如何修补实例的 SSM 文档概述](#) (p. 215) 和 [关于 SSM 文档 AWS-RunPatchBaseline](#) (p. 217)。

区别	详细信息
补丁评估	<p>Patch Manager 使用不同的流程评估应该在 Windows 托管实例和 Linux 托管实例上出现的补丁。对于 Windows 修补，Systems Manager 直接在服务中评估补丁基准规则以及批准和拒绝的补丁列表。它可以执行此操作是因为 Windows 补丁从单个存储库 (Windows 更新) 拉取。</p> <p>对于 Linux 修补，Systems Manager 在每个托管实例中评估补丁基准规则以及批准和拒绝的补丁列表。Systems Manager 必须在每个实例上评估修补，因为该服务从实例上配置的存储库检索已知补丁和更新列表。</p>
不适用的补丁	<p>由于 Linux 操作系统有大量可用的软件包，因此 Systems Manager 不报告处于不适用状态的补丁的详细信息。例如，当实例未安装 Apache 时，Apache 软件的补丁就是不适用的补丁。Systems Manager 会在摘要中报告不适用的补丁数量，但是如果您调用实例的 DescribeInstancePatches API，则返回的数据不包括处于不适用状态的补丁。这一行为不同于 Windows。</p>

已批准补丁和已拒绝补丁列表的程序包名称格式

可以添加到已批准补丁和已拒绝补丁列表中的程序包名称格式取决于您正在修补的操作系统类型。

适用于 Windows 操作系统的程序包名称格式

对于 Windows 操作系统，请使用 Microsoft 知识库 ID 和 Microsoft 安全公告 ID 指定补丁；例如：

```
KB2032276,KB2124261,MS10-048
```

适用于 Linux 操作系统的程序包名称格式

您可以在补丁基准中为已批准和已拒绝补丁指定的格式因 Linux 类型而异。更具体地说，支持的格式取决于 Linux 操作系统类型使用的程序包管理器。

主题

- [Amazon Linux、Amazon Linux 2、Red Hat Enterprise Linux \(RHEL\) 和 CentOS \(p. 223\)](#)
- [Ubuntu Server \(p. 223\)](#)
- [SUSE Linux Enterprise Server \(SLES\) \(p. 224\)](#)

Amazon Linux、Amazon Linux 2、Red Hat Enterprise Linux (RHEL) 和 CentOS

程序包管理器：YUM

已批准的补丁：对于已批准的补丁，您可以指定以下任一项：

- Bugzilla ID，格式为 1234567 (将只包含系统进程编号的字符串作为 Bugzilla ID。)
- CVE ID，格式为 CVE-2018-1234567
- 公告 ID，格式为 RHSA-2017:0864 和 ALAS-2018-123
- 完整程序包名称，格式为：
 - `example-pkg-0.710.10-2.7.abcd.x86_64`
 - `pkg-example-EE-20180914-2.2.amzn1.noarch`
- 包含单个通配符的程序包名称，格式为：
 - `example-pkg-*.abcd.x86_64`
 - `example-pkg-*-20180914-2.2.amzn1.noarch`
 - `example-pkg-EE-2018*.amzn1.noarch`

已拒绝的补丁：对于已拒绝的补丁，您可以指定以下任一项：

- 完整程序包名称，格式为：
 - `example-pkg-0.710.10-2.7.abcd.x86_64`
 - `pkg-example-EE-20180914-2.2.amzn1.noarch`
- 包含单个通配符的程序包名称，格式为：
 - `example-pkg-*.abcd.x86_64`
 - `example-pkg-*-20180914-2.2.amzn1.noarch`
 - `example-pkg-EE-2018*.amzn1.noarch`

Ubuntu Server

程序包管理器：APT

已批准的补丁和已拒绝的补丁：对于已批准和已拒绝的补丁，指定以下内容：

- 程序包名称，格式为 `ExamplePkg33`

Note

对于 Ubuntu Server 列表，不要包括架构或版本等元素。例如，您可以指定程序包名称 `ExamplePkg33` 以在补丁列表中包含所有以下项：

- `ExamplePkg33.x86.1`
- `ExamplePkg33.x86.2`

- `ExamplePkg33.x64.1`
- `ExamplePkg33.3.2.5-364.noarch`

SUSE Linux Enterprise Server (SLES)

程序包管理器：Zypper

已批准的补丁和已拒绝的补丁：对于已批准和已拒绝的补丁列表，可以指定以下任一项：

- 完整程序包名称，格式为：
 - `SUSE-SLE-Example-Package-12-2018-123`
 - `example-pkg-2018.11.4-46.17.1.x86_64.rpm`
- 包含单个通配符的程序包名称，例如：
 - `SUSE-SLE-Example-Package-12-2018-*`
 - `example-pkg-2018.11.4-46.17.1.*.rpm`

将实例组织到补丁组中

补丁组 是一种组织实例以进行修补的可选方法。例如，您可以为不同操作系统 (Linux 或 Windows)、不同环境 (开发、测试和生产) 或不同服务器功能 (Web 服务器、文件服务器、数据库) 创建补丁组。补丁组可以帮助您避免将补丁部署到一组错误的实例。它们还可帮助您避免过早地部署补丁 (在对补丁进行充分测试之前)。

您可以使用 Amazon EC2 标签创建补丁组。与其他跨 Systems Manager 的标记方案不同，必须使用此标签键定义补丁组：补丁组。请注意，此键区分大小写。您可以指定任何值，例如“Web 服务器”，但键必须为补丁组。

Note

一个实例只能在一个补丁组中。

创建补丁组和标记实例后，可以将补丁组注册到补丁基准。通过将补丁组注册到补丁基准，您可以确保在修补执行期间安装正确的补丁。

当系统运行将补丁基准应用于实例的任务时，服务将检查以了解是否为该实例定义了补丁组。如果该实例已分配给一个补丁组，则系统将检查以了解已将哪个补丁基准注册到该组。如果找到了该组的补丁基准，则系统将应用此补丁基准。如果没有为补丁组配置实例，系统将自动使用当前配置的默认补丁基准。

例如，假设有一个带有 `key=Patch Group` 和 `value=Front-End Servers` 标记的实例。当 Patch Manager 在该实例上运行 `AWS-RunPatchBaseline` 任务时，此服务将查看哪个补丁基准已注册到前端服务器。如果找到一个补丁基准，系统将使用该基准。如果未为前端服务器注册补丁基准，系统将使用默认补丁基准。

要查看使用 AWS CLI 创建补丁基准和补丁组的示例，请参阅[演练：修补服务器环境 \(AWS CLI\) \(p. 229\)](#)。有关 Amazon EC2 标签的更多信息，请参阅 Amazon EC2 用户指南 中的[标记 Amazon EC2 资源](#)。

使用 Maintenance Window 计划补丁更新

在配置补丁基准 (还可以选择配置补丁组) 后，您可以使用 Maintenance Window 将补丁应用于您的实例。Maintenance Window 通过让您指定在不中断业务运营的时间执行修补流程，可以减少对服务器可用性的影响。Maintenance Window 的工作方式如下所示：

1. 创建带修补操作计划的 Maintenance Window。
2. 通过为标签名称指定补丁组标签并指定已定义 Amazon EC2 标签的任意值 (例如，“生产服务器”)，来选择 Maintenance Window 的目标。

3. 创建新的 Maintenance Window 任务，并指定 AWS-RunPatchBaseline 文档。

在配置任务时，可以选择扫描实例，也可以选择扫描实例并安装补丁。如果选择扫描实例，则 Patch Manager 将扫描每个实例并生成缺失补丁的列表供您查看。

如果选择扫描并安装补丁，则 Patch Manager 将扫描每个实例并将已安装补丁的列表与基准中已批准补丁的列表进行比较。Patch Manager 将标识缺失的补丁，然后下载并安装所有缺失的补丁和已批准的补丁。

如果需要执行一次性扫描或安装来解决问题，可以使用 Run Command 直接调用 AWS-RunPatchBaseline 文档。

Important

在安装补丁后，Systems Manager 重启每个实例。需要重启才能确保正确安装补丁，并确保系统不会使实例处于潜在的不良状态。

Systems Manager Patch Manager 演练

以下演练说明如何使用 Systems Manager 控制台和 AWS CLI 创建补丁基准、补丁组和执行修补的 Maintenance Window。

内容

- [演练：修补服务器环境 \(控制台\) \(p. 225\)](#)
- [演练：修补服务器环境 \(AWS CLI\) \(p. 229\)](#)

演练：修补服务器环境 (控制台)

以下演练介绍如何在 Systems Manager 控制台中使用默认补丁基准、补丁组和 Maintenance Window 修补服务器环境。要详细了解此演练介绍的过程，请参阅 [使用 Patch Manager \(p. 219\)](#)。

在您开始之前

在您的实例上安装或更新 SSM 代理。要修补 Linux 实例，您的实例必须运行 SSM 代理 2.0.834.0 版或更高版本。有关更新此代理的信息，请参阅 [从控制台运行命令 \(p. 182\)](#) 中标题为示例：更新 SSM 代理的部分。

此外，以下演练在 Maintenance Window 期间运行修补程序。您必须先为 Maintenance Window 配置角色和权限，然后才能开始使用。有关更多信息，请参阅 [控制对 Maintenance Window 的访问权限 \(p. 247\)](#)。

主题

- [创建默认补丁基准 \(p. 225\)](#)
- [将实例添加到补丁组 \(p. 227\)](#)
- [创建 Maintenance Window 用于修补 \(p. 227\)](#)

创建默认补丁基准

Patch Manager 包含用于 Patch Manager 支持的每个操作系统的默认补丁基准。您可以利用这些默认补丁基准 (您不能对其进行自定义)，也可创建自己的补丁基准。以下过程介绍了如何查看默认补丁基准，以查看它们是否满足您的需求。该程序还介绍了如何创建自己的默认补丁基准。要了解有关补丁基准的更多信息，请参阅 [验证默认补丁基准或创建自定义补丁基准 \(p. 220\)](#)。

创建默认补丁基准 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。

2. 在导航窗格中，选择 Patch Manager。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Patch Manager。

3. 在补丁基准列表中，选择要修补的操作系统的补丁基准的名称。
4. 选择批准规则选项卡。

如果实例接受自动批准规则，可跳至下一过程 [将实例添加到补丁组 \(p. 227\)](#)。

-或者-

要创建自己的默认补丁基准，请在导航窗格中，选择 Patch Manager，然后选择创建补丁基准。

5. 在名称字段中，键入新补丁基准的名称，例如 **RHEL-Default**。
6. (可选) 键入此补丁基准的描述。
7. 在操作系统列表中，选择操作系统，例如 Red Hat Enterprise Linux。
8. 在批准规则部分，使用字段创建一个或多个自动批准规则。
 - 产品：批准规则适用于的操作系统的版本，例如 RedhatEnterpriseLinux7.4。默认选择为##。
 - 分类：批准规则适用于的补丁的类型，例如###。默认选择为##。
 - 严重性：规则适用于的补丁的严重性值，例如##。默认选择为##。
 - 自动批准延迟：发布补丁后到自动批准补丁前等待的天数。可以输入零 (0) 到 100 的任何整数。
 - (可选) 合规性级别：要分配给基准批准的补丁的严重性级别，例如#。

Note

如果有已批准的补丁报告为缺失，则在合规性级别中选择的选项 (如 Critical 或 Medium) 将确定违反合规性的严重性。

- (仅限 Linux) 包括除安全性之外的更新：选中此复选框，除了可以安装与安全性相关的补丁外，还可以安装源存储库中提供的非安全性补丁。

Note

对于 SUSE Linux Enterprise Server，无需选中此复选框，因为 SLES 实例上默认安装针对安全性和非安全性问题的补丁。有关更多信息，请参阅[如何选择安全性补丁 \(p. 205\)](#)中的 SLES 的内容。

有关在自定义补丁基准中使用批准规则的更多信息，请参阅[自定义基准 \(p. 221\)](#)。

9. 在补丁异常部分中，输入要为基准明确批准和拒绝的补丁列表 (使用逗号分隔)。对于已批准的补丁，选择相应的合规性严重性级别。

Note

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式 \(p. 222\)](#)。

如果您指定的任何已批准的补丁与安全性无关，请选中已批准的补丁包括除安全性之外的更新框以也安装这些补丁。仅适用于 Linux 实例。

10. (可选) 仅对于 Linux 实例：如果您要为不同版本的操作系统 (如 AmazonLinux2016.03 和 AmazonLinux2017.09) 指定备用补丁存储库，请为补丁来源部分中的每个产品执行以下操作：
 - 在名称中，键入名称以帮助您识别源配置。
 - 在产品中，选择补丁源存储库适用于的操作系统的版本，如 RedhatEnterpriseLinux7.4。
 - 在配置中，输入要使用的 yum 存储库配置的值。例如：

```
cachedir=/var/cache/yum/$basearch  
$releasever  
keepcache=0  
debuglevel=2
```

选择添加其他来源来为每个其他操作系统版本指定源存储库，最多 20 个。

有关备用源补丁存储库的更多信息，请参阅[如何指定备用补丁源存储库 \(Linux\) \(p. 208\)](#)。

11. 选择创建补丁基准。
12. 在补丁基准列表中，选择要设置为默认的基准。
13. 选择操作，然后选择设置默认补丁基准。
14. 验证设置默认补丁基准确认对话框中的详细信息，然后选择设置为默认。

将实例添加到补丁组

为了帮助您组织修补工作，我们建议您使用 Amazon EC2 标签将实例添加到补丁组。补丁组需要使用标签键 Patch Group。您可以指定任何值，但标签键必须为补丁组。有关补丁组的更多信息，请参阅[将实例组织到补丁组中 \(p. 224\)](#)。

将实例添加到补丁组

1. 打开 [Amazon EC2 控制台](#)，然后在左侧导航窗格中选择实例。
2. 从实例列表中选择您要配置用于修补的实例。
3. 从 Actions 菜单中，选择 Instance Settings、Add/Edit Tags。
4. 如果实例已应用一个或多个标签，请选择创建标签。
5. 在键字段中，键入 **Patch Group**。
6. 在 Value 字段中，键入可帮助您了解要修补的实例的值。
7. 选择 Save。
8. 重复此过程，向同一补丁组中添加其他实例。

创建Maintenance Window用于修补

为了最大程度减少对服务器可用性的影响，我们建议您将Maintenance Window配置为在不中断业务运营的时间执行修补。有关Maintenance Window的更多信息，请参阅[AWS Systems Manager Maintenance Window \(p. 246\)](#)。

创建 Maintenance Window 用于修补 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Maintenance Windows。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Maintenance Windows。

3. 选择 Create maintenance window。
4. 在 Name 字段中，键入一个将此项指定为用于修补关键更新和重要更新的Maintenance Window的名称。
5. 在计划部分顶部，选择所需的计划选项。

6. 在 Duration 字段中，键入您希望Maintenance Window处于活动状态的小时数。
7. 在 Stop initiating tasks 字段中，键入您希望系统在Maintenance Window周期结束前几小时停止启动新任务。
8. 选择 Create maintenance window。
9. 在 Maintenance Window 列表中，选择刚才创建的 Maintenance Window，然后选择操作、注册目标。
10. (可选) 在 Maintenance window target details 部分中，为此目标提供名称、描述和所有者信息 (您的姓名或别名)。
11. 在目标部分中，选择指定标签。
12. 在标签下，输入标签键和标签值来指定要注册 Maintenance Window 的实例。
13. 选择注册目标。系统将创建Maintenance Window目标。
14. 在创建的 Maintenance Window 的详细信息页面中，选择操作、注册运行命令任务。
15. (可选) 在 Maintenance window task details 部分中，为此任务提供名称和描述。
16. 在命令文档列表中，选择 AWS-RunPatchBaseline。
17. 在任务优先级列表中，选择优先级。1 表示最高优先级。
18. 在目标部分中定位方式的下方，选择在此过程中先前创建的 Maintenance Window 目标。
19. (可选) 在 Rate control 中：

- 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
20. 在角色部分中，输入附加 AmazonSSMMaintenanceWindowRole 的 IAM 角色的 ARN。有关更多信息，请参阅 [控制对 Maintenance Window 的访问权限 \(p. 247\)](#)。
 21. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

22. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

23. 在参数部分中：

- 在 Operation 列表中，选择 Scan 以扫描缺失的补丁，或选择 Install 以扫描并安装缺失的补丁。

Note

Install 操作将使实例重启 (如果已安装补丁)。Scan 操作不会导致重启。

- 您不需要在 Snapshot Id 字段中指定任何值。此系统将自动生成并提供此参数。
- (可选) 在注释框中，输入有关此命令的跟踪备注或提醒。
- 在“超时 (秒)”框中，输入系统在认为操作失败前等待操作完成的秒数。

24. 选择注册运行命令任务。

Maintenance Window任务完成后，您可以在 Managed Instances 页面上的 Amazon EC2 控制台中查看补丁合规性详细信息。在筛选栏中，使用 AWS:PatchSummary 和 AWS:ComplianceItem 筛选器。

Note

指定筛选器后，您可以通过为 URL 添加书签来保存您的查询。

您还可以通过在 Managed Instances 页面中选择实例来向下钻取到特定实例，然后选择 Patch 选项卡。您也可以使用 [DescribePatchGroupState](#) 和 [DescribeInstancePatchStatesForPatchGroup](#) API 来查看合规性详细信息。有关补丁合规性数据的信息，请参阅[关于补丁合规性 \(p. 95\)](#)。

演练：修补服务器环境 (AWS CLI)

以下过程将说明用户如何使用自定义补丁基准、补丁组和Maintenance Window来修补服务器环境。

有关可以用于 Patch Manager 配置任务的其他 AWS CLI 命令的示例，请参阅[Patch Manager 的 AWS CLI 命令 \(p. 234\)](#)。

在您开始之前

在您的实例上安装或更新 SSM 代理。要修补 Linux 实例，您的实例必须运行 SSM 代理 2.0.834.0 版或更高版本。有关更新此代理的信息，请参阅[从控制台运行命令 \(p. 182\)](#) 中标题为示例：更新 SSM 代理的部分。

此外，以下演练在 Maintenance Window 期间运行修补程序。您必须先为Maintenance Window配置角色和权限，然后才能开始使用。有关更多信息，请参阅[控制对 Maintenance Window 的访问权限 \(p. 247\)](#)。

使用 AWS CLI 配置 Patch Manager 和修补实例

1. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. (Windows) 执行以下命令以创建一个名为“Production-Baseline”的补丁基准，该补丁基准将在发布适用于生产环境的补丁 7 天后批准这些补丁。

```
aws ssm create-patch-baseline --name "Production-Baseline" --operating-
system "WINDOWS" --product "WindowsServer2012R2" --approval-rules
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important]}},
{Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,UpdateRollups,CriticalUpdates]}]},ApproveAfterD
  --description "Baseline containing all updates approved for production systems"
```

(Linux) 执行以下命令以创建一个名为“Production-Baseline”的补丁基准，该补丁基准将在发布适用于生产环境的补丁 7 天后批准这些补丁，包括源存储库中包括的安全性和非安全性补丁。

```
aws ssm create-patch-baseline --name "Production-Baseline"
  --operating-system "AMAZON_LINUX" --approval-rules
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=PRODUCT,Values=[AmazonLinux2016.03,AmazonLinux2
```



```
{Key=SEVERITY,Values=[Critical,Important]},  
{Key=CLASSIFICATION,Values=[Security]}}},ApproveAfterDays=7,EnableNonSecurity=true}}]"  
  --description "Baseline containing all updates approved for production systems"
```

系统将返回类似于以下内容的信息。

```
{  
  "BaselineId":"pb-0c10e65780EXAMPLE"  
}
```

4. 执行以下命令，为三个分别名为“Production”、“Database Servers”和“Front-End Patch Group”的补丁组注册“Production-Baseline”补丁基准。

```
aws ssm register-patch-baseline-for-patch-group --baseline-id pb-0c10e65780EXAMPLE" --  
patch-group "Production"
```

系统将返回类似于以下内容的信息。

```
{  
  "PatchGroup":"Production",  
  "BaselineId":"pb-0c10e65780EXAMPLE"  
}
```

```
aws ssm register-patch-baseline-for-patch-group --baseline-id pb-0c10e65780EXAMPLE" --  
patch-group "Database Servers"
```

系统将返回类似于以下内容的信息。

```
{  
  "PatchGroup":"Database Servers",  
  "BaselineId":"pb-0c10e65780EXAMPLE"  
}
```

5. 执行以下命令，为生产服务器创建两个Maintenance Window。第一个时段在每周二晚上 10 点运行。第二个时段在每周六晚上 10 点运行。

```
aws ssm create-maintenance-window --name "Production-Tuesdays" --schedule "cron(0 0  
22 ? * TUE *)" --duration 1 --cutoff 0 --no-allow-unassociated-targets
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowId":"mw-0c66948c711a3b5bd"  
}
```

```
aws ssm create-maintenance-window --name "Production-Saturdays" --schedule "cron(0 0  
22 ? * SAT *)" --duration 2 --cutoff 0 --no-allow-unassociated-targets
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowId":"mw-09e2a75baadd84e85"  
}
```

6. 执行以下命令，将生产服务器注册到两个生产Maintenance Window。

```
aws ssm register-target-with-maintenance-window --window-id mw-0c66948c711a3b5bd  
--targets "Key=tag:Patch Group,Values=Production" --owner-information "Production  
servers" --resource-type "INSTANCE"
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTargetId": "557e7b3a-bc2f-48dd-ae05-e282b5b20760"  
}
```

```
aws ssm register-target-with-maintenance-window --window-id mw-0c66948c711a3b5bd --  
targets "Key=tag:Patch Group,Values=Database Servers" --owner-information "Database  
servers" --resource-type "INSTANCE"
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTargetId": "767b6508-f4ac-445e-b6fe-758cc912e55c"  
}
```

```
aws ssm register-target-with-maintenance-window --window-id mw-09e2a75baadd84e85  
--targets "Key=tag:Patch Group,Values=Production" --owner-information "Production  
servers" --resource-type "INSTANCE"
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTargetId": "faa01c41-1d57-496c-ba77-ff9cadba4b7d"  
}
```

```
aws ssm register-target-with-maintenance-window --window-id mw-09e2a75baadd84e85 --  
targets "Key=tag:Patch Group,Values=Database Servers" --owner-information "Database  
servers" --resource-type "INSTANCE"
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTargetId": "673b5840-58a4-42ab-8b80-95749677cb2e"  
}
```

7. 执行以下命令以注册一个补丁任务，此任务仅在第一个生产Maintenance Window内扫描生产服务器是否存在缺失的更新。

```
aws ssm register-task-with-maintenance-window --window-id mw-0c66948c711a3b5bd --  
targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-ae05-e282b5b20760" --task-arn  
"AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"  
--task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 1 --task-  
parameters '{"Operation\":[{"Values\":[{"Scan"}]}'
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTaskId": "968e3b17-8591-4fb2-932a-b62389d6f635"  
}
```



```
}
```

```
aws ssm register-task-with-maintenance-window --window-id mw-0c66948c711a3b5bd --  
targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" --task-arn  
"AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"  
--task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 5 --task-  
parameters '{"Operation\":"Values\":[\"Scan\"]}]}'
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTaskId":"09f2e873-a3a7-443f-ba0a-05cf4de5a1c7"  
}
```

8. 执行以下命令以注册一个补丁任务，此任务在第二个Maintenance Window内在生产服务器上安装缺失的更新。

```
aws ssm register-task-with-maintenance-window --window-id mw-09e2a75baadd84e85 --  
targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-ae05-e282b5b20760" --task-arn  
"AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"  
--task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 1 --task-  
parameters '{"Operation\":"Values\":[\"Install\"]}]}'
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTaskId":"968e3b17-8591-4fb2-932a-b62389d6f635"  
}
```

```
aws ssm register-task-with-maintenance-window --window-id mw-09e2a75baadd84e85 --  
targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" --task-arn  
"AWS-ApplyPatchBaseline" --service-role-arn "arn:aws:iam::12345678:role/MW-Role"  
--task-type "RUN_COMMAND" --max-concurrency 2 --max-errors 1 --priority 5 --task-  
parameters '{"Operation\":"Values\":[\"Install\"]}]}'
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowTaskId":"09f2e873-a3a7-443f-ba0a-05cf4de5a1c7"  
}
```

9. 执行以下命令以获取补丁组的高级补丁合规性摘要。高级补丁合规性摘要将为您提供补丁组的大量具有以下状态的补丁的实例数：“NotApplicable”、“Missing”、“Failed”、“InstalledOther”和“Installed”。

```
aws ssm describe-patch-group-state --patch-group "Production"
```

系统将返回类似于以下内容的信息。

```
{  
  "InstancesWithNotApplicablePatches":0,  
  "InstancesWithMissingPatches":0,  
  "InstancesWithFailedPatches":1,  
  "InstancesWithInstalledOtherPatches":4,  
  "Instances":4,  
  "InstancesWithInstalledPatches":3  
}
```

10. 执行以下命令以获取补丁组的每个实例的补丁摘要状态。每个实例的摘要将为您提供补丁组的每个实例中具有以下状态的大量补丁：“NotApplicable”、“Missing”、“Failed”、“InstalledOther”和“Installed”。

```
aws ssm describe-instance-patch-states-for-patch-group --patch-group "Production"
```

系统将返回类似于以下内容的信息。

```
{
  "InstancePatchStates":[
    {
      "OperationStartTime":1481259600.0,
      "FailedCount":0,
      "InstanceId":"i-08ee91c0b17045407",
      "OwnerInformation":"",
      "NotApplicableCount":2077,
      "OperationEndTime":1481259757.0,
      "PatchGroup":"Production",
      "InstalledOtherCount":186,
      "MissingCount":7,
      "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation":"Scan",
      "InstalledCount":72
    },
    {
      "OperationStartTime":1481259602.0,
      "FailedCount":0,
      "InstanceId":"i-0fff3aab684d01b23",
      "OwnerInformation":"",
      "NotApplicableCount":2692,
      "OperationEndTime":1481259613.0,
      "PatchGroup":"Production",
      "InstalledOtherCount":3,
      "MissingCount":1,
      "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation":"Scan",
      "InstalledCount":1
    },
    {
      "OperationStartTime":1481259547.0,
      "FailedCount":0,
      "InstanceId":"i-0a00def7faa94f1dc",
      "OwnerInformation":"",
      "NotApplicableCount":1859,
      "OperationEndTime":1481259592.0,
      "PatchGroup":"Production",
      "InstalledOtherCount":116,
      "MissingCount":1,
      "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation":"Scan",
      "InstalledCount":110
    },
    {
      "OperationStartTime":1481259549.0,
      "FailedCount":0,
      "InstanceId":"i-09a618aec652973a9",
      "OwnerInformation":"",
      "NotApplicableCount":1637,
      "OperationEndTime":1481259837.0,
      "PatchGroup":"Production",
      "InstalledOtherCount":388,
      "MissingCount":2,
      "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",
      "Operation":"Scan",
```

```
        "InstalledCount":141
      }
    ]
  }
}
```

Patch Manager 的 AWS CLI 命令

此部分包括您可用于执行 Patch Manager 配置任务的 CLI 命令示例。

有关通过 AWS CLI 使用自定义补丁基准对服务器环境进行修补的说明，请参阅 [演练：修补服务器环境 \(AWS CLI\)](#) (p. 229)。

有关使用 CLI 执行 AWS Systems Manager 任务的更多信息，请参阅 [AWS Systems Manager section of the AWS CLI Command Reference](#)。

示例命令

- [创建补丁基准](#) (p. 234)
- [创建对不同操作系统版本使用自定义存储库的补丁基准](#) (p. 235)
- [更新补丁基准](#) (p. 236)
- [重命名补丁基准](#) (p. 237)
- [删除补丁基准](#) (p. 238)
- [列出所有补丁基准](#) (p. 238)
- [列出所有 AWS 提供的补丁基准](#) (p. 239)
- [列出我的补丁基准](#) (p. 239)
- [显示补丁基准](#) (p. 239)
- [获取默认补丁基准](#) (p. 240)
- [设置默认补丁基准](#) (p. 241)
- [将补丁组“Web 服务器”注册到补丁基准](#) (p. 241)
- [将补丁组“后端”注册到 AWS 提供的补丁基准](#) (p. 241)
- [显示补丁组注册](#) (p. 241)
- [从补丁基准取消注册补丁组](#) (p. 242)
- [获取补丁基准定义的所有补丁](#) (p. 242)
- [获取所有 MSRC 严重性为“关键”的适用于 Windows Server 2012 的补丁](#) (p. 243)
- [获取所有可用补丁](#) (p. 243)
- [标记补丁基准](#) (p. 245)
- [列出补丁基准的标签](#) (p. 245)
- [从补丁基准删除标签](#) (p. 245)
- [获取每个实例的修补摘要状态](#) (p. 245)
- [获取实例的补丁合规性详细信息](#) (p. 246)

创建补丁基准

以下命令将创建一个补丁基准，该补丁基准将在发布 Windows Server 2012 R2 的关键和重要安全更新后 5 天内批准所有这些更新。

```
aws ssm create-patch-baseline --name "Windows-Server-2012R2" --approval-rules
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Important,Critical]}},
{Key=CLASSIFICATION,Values=SecurityUpdates},
{Key=PRODUCT,Values=WindowsServer2012R2}]],ApproveAfterDays=5}]" --description "Windows
  Server 2012 R2, Important and Critical security updates"
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

创建对不同操作系统版本使用自定义存储库的补丁基准

仅适用于 Linux 实例。以下命令显示如何指定要用于特定版本的 Amazon Linux 操作系统的补丁存储库。此示例使用 Amazon Linux 2017.09 上默认启用的源存储库，但可以调整为为您为实例配置的不同源存储库。

Note

为了更好地演示此更为复杂的命令，我们使用 `-cli-input-json` 选项与存储在外部 JSON 文件中的其他选项。

1. 创建一个名称类似于 `my-patch-repository.json` 的 JSON 文件并将以下内容添加到其中：

```
{
  "Description": "My patch repository for Amazon Linux 2017.09",
  "Name": "Amazon-Linux-2017.09",
  "OperatingSystem": "AMAZON_LINUX",
  "ApprovalRules": {
    "PatchRules": [
      {
        "ApproveAfterDays": 7,
        "EnableNonSecurity": true,
        "PatchFilterGroup": {
          "PatchFilters": [
            {
              "Key": "SEVERITY",
              "Values": [
                "Important",
                "Critical"
              ]
            },
            {
              "Key": "CLASSIFICATION",
              "Values": [
                "Security",
                "Bugfix"
              ]
            },
            {
              "Key": "PRODUCT",
              "Values": [
                "AmazonLinux2017.09"
              ]
            }
          ]
        }
      }
    ]
  },
  "Sources": [
    {
      "Name": "My-AL2017.09",
      "Products": [
        "AmazonLinux2017.09"
      ],
      "Configuration": "[amzn-main] \nname=amzn-main-Base\nnmirrorlist=http://repo.$awsregion.$awsdomain/$releasever/main/mirror.list //nmirrorlist_expire=300//nmetadata_expire=300 \npriority=10 \nfailovermethod=priority \nfatestmirror_enabled=0"
    }
  ]
}
```

```
\ngpgcheck=1 \ngpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-ga \nenabled=1
\nretries=3 \ntimeout=5\nreport_instanceid=yes"
    }
  ]
}
```

2. 在保存文件的目录中，键入以下命令：

```
aws ssm create-patch-baseline --cli-input-json file://my-patch-repository.json
```

系统将返回类似于以下内容的信息：

```
{
  "BaselineId": "pb-12343b962ba63wxya"
}
```

更新补丁基准

以下命令将两个带已拒绝状态的补丁和一个带已批准状态的补丁添加到现有补丁基准。

Note

有关批准的修补程序和拒绝的修补程序列表的接受格式的信息，请参阅 [已批准补丁和已拒绝补丁列表的程序包名称格式](#) (p. 222)。

```
aws ssm update-patch-baseline --baseline-id pb-0c10e65780EXAMPLE --rejected-patches
"KB2032276" "MS10-048" --approved-patches "KB2124261"
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineId": "pb-0c10e65780EXAMPLE",
  "Name": "Windows-Server-2012R2",
  "RejectedPatches": [
    "KB2032276",
    "MS10-048"
  ],
  "GlobalFilters": {
    "PatchFilters": [
    ]
  },
  "ApprovalRules": {
    "PatchRules": [
      {
        "PatchFilterGroup": {
          "PatchFilters": [
            {
              "Values": [
                "Important",
                "Critical"
              ],
              "Key": "MSRC_SEVERITY"
            },
            {
              "Values": [
                "SecurityUpdates"
              ],
              "Key": "CLASSIFICATION"
            }
          ]
        }
      }
    ]
  }
}
```

```

        },
        {
            "Values": [
                "WindowsServer2012R2"
            ],
            "Key": "PRODUCT"
        }
    ],
    },
    "ApproveAfterDays": 5
}
]
},
"ModifiedDate": 1481001494.035,
"CreatedDate": 1480997823.81,
"ApprovedPatches": [
    "KB2124261"
],
"Description": "Windows Server 2012 R2, Important and Critical security updates"
}

```

重命名补丁基准

```
aws ssm update-patch-baseline --baseline-id pb-0c10e65780EXAMPLE" --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"
```

系统将返回类似于以下内容的信息。

```

{
  "BaselineId": "pb-0c10e65780EXAMPLE",
  "Name": "Windows-Server-2012-R2-Important-and-Critical-Security-Updates",
  "RejectedPatches": [
    "KB2032276",
    "MS10-048"
  ],
  "GlobalFilters": {
    "PatchFilters": [
    ]
  },
  "ApprovalRules": {
    "PatchRules": [
      {
        "PatchFilterGroup": {
          "PatchFilters": [
            {
              "Values": [
                "Important",
                "Critical"
              ],
              "Key": "MSRC_SEVERITY"
            },
            {
              "Values": [
                "SecurityUpdates"
              ],
              "Key": "CLASSIFICATION"
            },
            {
              "Values": [
                "WindowsServer2012R2"
              ],
              "Key": "PRODUCT"
            }
          ]
        }
      }
    ]
  }
}

```

```
        }
      ],
    },
    "ApproveAfterDays":5
  }
],
},
"ModifiedDate":1481001795.287,
"CreateDate":1480997823.81,
"ApprovedPatches":[
  "KB2124261"
],
"Description":"Windows Server 2012 R2, Important and Critical security updates"
}
```

删除补丁基准

```
aws ssm delete-patch-baseline --baseline-id "pb-0c10e65780EXAMPLE"
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineId":"pb-0c10e65780EXAMPLE"
}
```

列出所有补丁基准

```
aws ssm describe-patch-baselines
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineIdentities":[
    {
      "BaselineName":"AWS-DefaultPatchBaseline",
      "DefaultBaseline":true,
      "BaselineDescription":"Default Patch Baseline Provided by AWS.",
      "BaselineId":"arn:aws:ssm:us-east-2:111122223333:patchbaseline/
pb-0c10e65780EXAMPLE"
    },
    {
      "BaselineName":"Windows-Server-2012R2",
      "DefaultBaseline":false,
      "BaselineDescription":"Windows Server 2012 R2, Important and Critical security
updates",
      "BaselineId":"pb-0c10e65780EXAMPLE"
    }
  ]
}
```

以下是另一个命令，该命令将列出区域中的所有补丁基准。

```
aws ssm describe-patch-baselines --region us-east-2 --filters "Key=OWNER,Values=[All]"
```

系统将返回类似于以下内容的信息。

```
{
```

```
"BaselineIdentities":[
  {
    "BaselineName":"AWS-DefaultPatchBaseline",
    "DefaultBaseline":true,
    "BaselineDescription":"Default Patch Baseline Provided by AWS.",
    "BaselineId":"arn:aws:ssm:us-east-2:111122223333:patchbaseline/
pb-0c10e65780EXAMPLE"
  },
  {
    "BaselineName":"Windows-Server-2012R2",
    "DefaultBaseline":false,
    "BaselineDescription":"Windows Server 2012 R2, Important and Critical security
updates",
    "BaselineId":"pb-0c10e65780EXAMPLE"
  }
]
```

列出所有 AWS 提供的补丁基准

```
aws ssm describe-patch-baselines --region us-east-2 --filters "Key=OWNER,Values=[AWS]"
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineIdentities":[
    {
      "BaselineName":"AWS-DefaultPatchBaseline",
      "DefaultBaseline":true,
      "BaselineDescription":"Default Patch Baseline Provided by AWS.",
      "BaselineId":"arn:aws:ssm:us-east-2:111122223333:patchbaseline/
pb-0c10e65780EXAMPLE"
    }
  ]
}
```

列出我的补丁基准

```
aws ssm describe-patch-baselines --region us-east-2 --filters "Key=OWNER,Values=[Self]"
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineIdentities":[
    {
      "BaselineName":"Windows-Server-2012R2",
      "DefaultBaseline":false,
      "BaselineDescription":"Windows Server 2012 R2, Important and Critical security
updates",
      "BaselineId":"pb-0c10e65780EXAMPLE"
    }
  ]
}
```

显示补丁基准

```
aws ssm get-patch-baseline --baseline-id pb-0c10e65780EXAMPLE"
```


系统将返回类似于以下内容的信息。

```
{
  "BaselineId": "pb-0c10e65780EXAMPLE",
  "Name": "Windows-Server-2012R2",
  "PatchGroups": [
    "Web Servers"
  ],
  "RejectedPatches": [
  ],
  "GlobalFilters": {
    "PatchFilters": [
    ]
  },
  "ApprovalRules": {
    "PatchRules": [
      {
        "PatchFilterGroup": {
          "PatchFilters": [
            {
              "Values": [
                "Important",
                "Critical"
              ],
              "Key": "MSRC_SEVERITY"
            },
            {
              "Values": [
                "SecurityUpdates"
              ],
              "Key": "CLASSIFICATION"
            },
            {
              "Values": [
                "WindowsServer2012R2"
              ],
              "Key": "PRODUCT"
            }
          ]
        },
        "ApproveAfterDays": 5
      }
    ]
  },
  "ModifiedDate": 1480997823.81,
  "CreatedDate": 1480997823.81,
  "ApprovedPatches": [
  ],
  "Description": "Windows Server 2012 R2, Important and Critical security updates"
}
```

获取默认补丁基准

```
aws ssm get-default-patch-baseline --region us-east-2
```

系统将返回类似于以下内容的信息。

```
{
  "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
```

```
}
```

设置默认补丁基准

```
aws ssm register-default-patch-baseline --region us-east-2 --baseline-id  
"pb-0c10e65780EXAMPLE"
```

```
{  
  "BaselineId": "pb-0c10e65780EXAMPLE"  
}
```

将补丁组“Web 服务器”注册到补丁基准

```
aws ssm register-patch-baseline-for-patch-group --baseline-id "pb-0c10e65780EXAMPLE" --  
patch-group "Web Servers"
```

系统将返回类似于以下内容的信息。

```
{  
  "PatchGroup": "Web Servers",  
  "BaselineId": "pb-0c10e65780EXAMPLE"  
}
```

将补丁组“后端”注册到 AWS 提供的补丁基准

```
aws ssm register-patch-baseline-for-patch-group --region us-east-2 --baseline-id  
"arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE" --patch-group  
"Backend"
```

系统将返回类似于以下内容的信息。

```
{  
  "PatchGroup": "Backend",  
  "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"  
}
```

显示补丁组注册

```
aws ssm describe-patch-groups --region us-east-2
```

系统将返回类似于以下内容的信息。

```
{  
  "PatchGroupPatchBaselineMappings": [  
    {  
      "PatchGroup": "Backend",  
      "BaselineIdentity": {  
        "BaselineName": "AWS-DefaultPatchBaseline",  
        "DefaultBaseline": false,  
        "BaselineDescription": "Default Patch Baseline Provided by AWS.",  
        "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/  
pb-0c10e65780EXAMPLE"  
      }  
    }  
  ]  
}
```

```
    },
    {
      "PatchGroup": "Web Servers",
      "BaselineIdentity": {
        "BaselineName": "Windows-Server-2012R2",
        "DefaultBaseline": true,
        "BaselineDescription": "Windows Server 2012 R2, Important and Critical updates",
        "BaselineId": "pb-0c10e65780EXAMPLE"
      }
    }
  ]
}
```

从补丁基准取消注册补丁组

```
aws ssm deregister-patch-baseline-for-patch-group --region us-east-2 --patch-group
"Production" --baseline-id "arn:aws:ssm:us-east-2:111122223333:patchbaseline/
pb-0c10e65780EXAMPLE"
```

系统将返回类似于以下内容的信息。

```
{
  "PatchGroup": "Production",
  "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
}
```

获取补丁基准定义的所有补丁

```
aws ssm describe-effective-patches-for-patch-baseline --region us-east-2 --baseline-id
"pb-0c10e65780EXAMPLE"
```

系统将返回类似于以下内容的信息。

```
{
  "NextToken": "--token string truncated--",
  "EffectivePatches": [
    {
      "PatchStatus": {
        "ApprovalDate": 1384711200.0,
        "DeploymentStatus": "APPROVED"
      },
      "Patch": {
        "ContentUrl": "https://support.microsoft.com/en-us/kb/2876331",
        "ProductFamily": "Windows",
        "Product": "WindowsServer2012R2",
        "Vendor": "Microsoft",
        "Description": "A security issue has been identified in a Microsoft software product that could affect your system. You can help protect your system by installing this update from Microsoft. For a complete listing of the issues that are included in this update, see the associated Microsoft Knowledge Base article. After you install this update, you may have to restart your system.",
        "Classification": "SecurityUpdates",
        "Title": "Security Update for Windows Server 2012 R2 Preview (KB2876331)",
        "ReleaseDate": 1384279200.0,
        "MsrcClassification": "Critical",
        "Language": "All",
        "KbNumber": "KB2876331",
        "MsrcNumber": "MS13-089",
        "Id": "e74ccc76-85f0-4881-a738-59e9fc9a336d"
      }
    }
  ]
}
```

```

    }
  },
  {
    "PatchStatus":{
      "ApprovalDate":1428858000.0,
      "DeploymentStatus":"APPROVED"
    },
    "Patch":{
      "ContentUrl":"https://support.microsoft.com/en-us/kb/2919355",
      "ProductFamily":"Windows",
      "Product":"WindowsServer2012R2",
      "Vendor":"Microsoft",
      "Description":"Windows Server 2012 R2 Update is a cumulative set of security
updates, critical updates and updates. You must install Windows Server 2012 R2 Update
to ensure that your computer can continue to receive future Windows Updates, including
security updates. For a complete listing of the issues that are included in this update,
see the associated Microsoft Knowledge Base article for more information. After you
install this item, you may have to restart your computer.",
      "Classification":"SecurityUpdates",
      "Title":"Windows Server 2012 R2 Update (KB2919355)",
      "ReleaseDate":1428426000.0,
      "MsrcClassification":"Critical",
      "Language":"All",
      "KbNumber":"KB2919355",
      "MsrcNumber":"MS14-018",
      "Id":"8452bac0-bf53-4fbd-915d-499de08c338b"
    }
  }
}
---output truncated---
```

获取所有 MSRC 严重性为“关键”的适用于 Windows Server 2012 的补丁

```
aws ssm describe-available-patches --region us-east-2 --filters
Key=PRODUCT,Values=WindowsServer2012 Key=MSRC_SEVERITY,Values=Critical
```

系统将返回类似于以下内容的信息。

```

{
  "Patches":[
    {
      "ContentUrl":"https://support.microsoft.com/en-us/kb/2727528",
      "ProductFamily":"Windows",
      "Product":"WindowsServer2012",
      "Vendor":"Microsoft",
      "Description":"A security issue has been identified that could allow an
unauthenticated remote attacker to compromise your system and gain control over it. You
can help protect your system by installing this update from Microsoft. After you install
this update, you may have to restart your system.",
      "Classification":"SecurityUpdates",
      "Title":"Security Update for Windows Server 2012 (KB2727528)",
      "ReleaseDate":1352829600.0,
      "MsrcClassification":"Critical",
      "Language":"All",
      "KbNumber":"KB2727528",
      "MsrcNumber":"MS12-072",
      "Id":"1eb507be-2040-4eeb-803d-abc55700b715"
    },
    {
      "ContentUrl":"https://support.microsoft.com/en-us/kb/2729462",
      "ProductFamily":"Windows",
      "Product":"WindowsServer2012",

```

```
        "Vendor": "Microsoft",
        "Description": "A security issue has been identified that could allow an
unauthenticated remote attacker to compromise your system and gain control over it. You
can help protect your system by installing this update from Microsoft. After you install
this update, you may have to restart your system.",
        "Classification": "SecurityUpdates",
        "Title": "Security Update for Microsoft .NET Framework 3.5 on Windows 8 and Windows
Server 2012 for x64-based Systems (KB2729462)",
        "ReleaseDate": "1352829600.0",
        "MsrcClassification": "Critical",
        "Language": "All",
        "KbNumber": "KB2729462",
        "MsrcNumber": "MS12-074",
        "Id": "af873760-c97c-4088-ab7e-5219e120eab4"
    }
}

---output truncated---
```

获取所有可用补丁

```
aws ssm describe-available-patches --region us-east-2
```

系统将返回类似于以下内容的信息。

```
{
  "NextToken": "--token string truncated--",
  "Patches": [
    {
      "ContentUrl": "https://support.microsoft.com/en-us/kb/2032276",
      "ProductFamily": "Windows",
      "Product": "WindowsServer2008R2",
      "Vendor": "Microsoft",
      "Description": "A security issue has been identified that could allow an
unauthenticated remote attacker to compromise your system and gain control over it. You
can help protect your system by installing this update from Microsoft. After you install
this update, you may have to restart your system.",
      "Classification": "SecurityUpdates",
      "Title": "Security Update for Windows Server 2008 R2 x64 Edition (KB2032276)",
      "ReleaseDate": "1279040400.0",
      "MsrcClassification": "Important",
      "Language": "All",
      "KbNumber": "KB2032276",
      "MsrcNumber": "MS10-043",
      "Id": "8692029b-a3a2-4a87-a73b-8ea881b4b4d6"
    },
    {
      "ContentUrl": "https://support.microsoft.com/en-us/kb/2124261",
      "ProductFamily": "Windows",
      "Product": "Windows7",
      "Vendor": "Microsoft",
      "Description": "A security issue has been identified that could allow an
unauthenticated remote attacker to compromise your system and gain control over it. You
can help protect your system by installing this update from Microsoft. After you install
this update, you may have to restart your system.",
      "Classification": "SecurityUpdates",
      "Title": "Security Update for Windows 7 (KB2124261)",
      "ReleaseDate": "1284483600.0",
      "MsrcClassification": "Important",
      "Language": "All",
      "KbNumber": "KB2124261",
      "MsrcNumber": "MS10-065",
      "Id": "12ef1bed-0dd2-4633-b3ac-60888aa8ba33"
    }
  ]
}
```

```
---output truncated---
```

标记补丁基准

```
aws ssm add-tags-to-resource --resource-type "PatchBaseline" --resource-id  
"pb-0c10e65780EXAMPLE" --tags "Key=Project,Value=Testing"
```

列出补丁基准的标签

```
aws ssm list-tags-for-resource --resource-type "PatchBaseline" --resource-id  
"pb-0c10e65780EXAMPLE"
```

从补丁基准删除标签

```
aws ssm remove-tags-from-resource --resource-type "PatchBaseline" --resource-id  
"pb-0c10e65780EXAMPLE" --tag-keys "Project"
```

获取每个实例的修补摘要状态

每个实例的摘要将为您提供每个实例中具有以下状态的大量补丁：
“NotApplicable”、“Missing”、“Failed”、“InstalledOther”和“Installed”。

```
aws ssm describe-instance-patch-states --instance-ids i-08ee91c0b17045407  
i-09a618aec652973a9 i-0a00def7faa94f1c i-0fff3aab684d01b23
```

系统将返回类似于以下内容的信息。

```
{  
  "InstancePatchStates":[  
    {  
      "OperationStartTime":"2016-12-09T05:00:00Z",  
      "FailedCount":0,  
      "InstanceId":"i-08ee91c0b17045407",  
      "OwnerInformation": "",  
      "NotApplicableCount":2077,  
      "OperationEndTime":"2016-12-09T05:02:37Z",  
      "PatchGroup":"Production",  
      "InstalledOtherCount":186,  
      "MissingCount":7,  
      "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",  
      "Operation":"Scan",  
      "InstalledCount":72  
    },  
    {  
      "OperationStartTime":"2016-12-09T04:59:09Z",  
      "FailedCount":0,  
      "InstanceId":"i-09a618aec652973a9",  
      "OwnerInformation": "",  
      "NotApplicableCount":1637,  
      "OperationEndTime":"2016-12-09T05:03:57Z",  
      "PatchGroup":"Production",  
      "InstalledOtherCount":388,  
      "MissingCount":2,  
      "SnapshotId":"b0e65479-79be-4288-9f88-81c96bc3ed5e",  
      "Operation":"Scan",  
      "InstalledCount":141  
    }  
  ]  
}
```

```
}  
---output truncated---
```

获取实例的补丁合规性详细信息

```
aws ssm describe-instance-patches --instance-id i-08ee91c0b17045407
```

系统将返回类似于以下内容的信息。

```
{  
  "NextToken": "--token string truncated--",  
  "Patches": [  
    {  
      "KBId": "KB2919355",  
      "Severity": "Critical",  
      "Classification": "SecurityUpdates",  
      "Title": "Windows 8.1 Update for x64-based Systems (KB2919355)",  
      "State": "Installed",  
      "InstalledTime": "2014-03-18T12:00:00Z"  
    },  
    {  
      "KBId": "KB2977765",  
      "Severity": "Important",  
      "Classification": "SecurityUpdates",  
      "Title": "Security Update for Microsoft .NET Framework 4.5.1 and 4.5.2 on Windows  
8.1 and Windows Server 2012 R2 x64-based Systems (KB2977765)",  
      "State": "Installed",  
      "InstalledTime": "2014-10-15T12:00:00Z"  
    },  
    {  
      "KBId": "KB2978126",  
      "Severity": "Important",  
      "Classification": "SecurityUpdates",  
      "Title": "Security Update for Microsoft .NET Framework 4.5.1 and 4.5.2 on Windows  
8.1 (KB2978126)",  
      "State": "Installed",  
      "InstalledTime": "2014-11-18T12:00:00Z"  
    },  
  ],  
  ---output truncated---  
}
```

AWS Systems Manager Maintenance Window

借助 AWS Systems Manager Maintenance Window，您可以制定计划，规定何时对实例执行有可能造成中断的操作，例如修补操作系统、更新驱动程序或安装软件或补丁。每一个 Maintenance Window 都有一个计划、一段持续时间、一组注册目标和一组注册任务。利用 Maintenance Window，您可以执行类似于下面的任务：

- 安装或更新应用程序。
- 应用补丁。
- 安装或更新 SSM 代理。
- 使用 Systems Manager Run Command 任务运行 PowerShell 命令和 Linux Shell 脚本。
- 使用 Systems Manager Automation 来构建 Amazon 系统映像 (AMI)、引导软件和配置实例。
- 运行触发其他操作 (例如扫描您的实例是否有补丁更新) 的 AWS Lambda 函数。
- 运行 AWS Step Functions 状态机以执行从 Elastic Load Balancing 环境中删除实例、修补实例，然后将实例添加回 Elastic Load Balancing 环境等任务。

内容

- [控制对 Maintenance Window 的访问权限 \(p. 247\)](#)
- [使用 Maintenance Window \(p. 255\)](#)
- [Systems Manager Maintenance Window 演练 \(p. 259\)](#)

控制对 Maintenance Window 的访问权限

您的账户中的用户必须先获得必要权限，然后才能创建和计划 Maintenance Window 任务。授予这些权限的过程包含两个任务：

1. 任务 1：向 Maintenance Window 服务提供在实例上运行 Maintenance Window 任务所需的 AWS Identity and Access Management (IAM) 权限。您可以通过在创建 Maintenance Window 任务时为 Maintenance Window 任务创建自定义服务角色，然后将此角色指定为配置的一部分来执行此操作。
2. 任务 2：向您账户中将向 Maintenance Window 分配任务的用户授予 `iam:PassRole` 权限。这将允许他们将角色传递给 Maintenance Window 服务。如果没有明确授予此权限，用户将无法向 Maintenance Window 分配任务。

主题

- [控制对 Maintenance Window 的访问 \(控制台\) \(p. 247\)](#)
- [控制对 Maintenance Window 的访问 \(AWS CLI\) \(p. 250\)](#)
- [控制对 Maintenance Window 的访问 \(Tools for Windows PowerShell\) \(p. 252\)](#)
- [排除 IAM Maintenance Window 权限故障 \(p. 255\)](#)

控制对 Maintenance Window 的访问 (控制台)

以下过程介绍如何使用 AWS Systems Manager 控制台来创建 Maintenance Window 所需的角色和权限。

主题

- [任务 1：为 Maintenance Window 创建自定义服务角色 \(p. 247\)](#)
- [任务 2：将 IAM PassRole 策略分配到 IAM 用户或组 \(p. 249\)](#)

任务 1：为 Maintenance Window 创建自定义服务角色

使用以下过程为 Maintenance Window 创建自定义服务角色，以便 Systems Manager 可以代表您运行任务。

创建自定义服务角色

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择 Roles，然后选择 Create Role。
3. 标记以下选择：
 1. Select type of trusted entity (选择受信任实体的类型) 区域：AWS 服务
 2. 选择将使用此角色的服务 区域：EC2
 3. 选择您的使用案例区域：EC2
4. 选择 Next: Permissions。
5. 在策略列表中，选中 AmazonSSMMaintenanceWindowRole 旁边的复选框，然后选择 Next: Review。

6. 在角色名称 中，输入将此角色标识为 Maintenance Window 角色的名称，例如 **my-maintenance-window-role**。
7. 可选：更改默认角色描述，以反映此角色的用途。例如：“代表您执行维护时段任务”。
8. 选择 Create role。系统将让您返回到 Roles 页。
9. 选择刚才创建的角色名称。
10. 选择 Trust relationships 选项卡，然后选择 Edit trust relationship。
11. 删除当前策略，然后将下面的策略复制并粘贴到 Policy Document 字段中：

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

Note

仅当要使用 Amazon SNS 发送与通过 Run Command 运行的 Maintenance Window 任务相关的通知时，才需要 "sns.amazonaws.com"。有关更多信息，请参阅步骤 13。

12. 选择 Update Trust Policy，然后复制或记下 Summary 页面上的角色名称和 Role ARN 值。当您创建 Maintenance Window 时，将要指定此信息。
13. 如果要 Maintenance Window 配置为在通过 Run Command 命令任务运行时使用 Amazon SNS 发送有关命令状态的通知，请执行以下操作：
 1. 选择 Permissions 选项卡。
 2. 选择 Add inline policy，然后选择 JSON 选项卡。
 3. 在 Policy Document 中，粘贴以下内容：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "sns-access-role-arn"
    }
  ]
}
```

sns-access-role-arn 表示用于发送与现有 Maintenance Window 相关的 SNS 通知的 IAM 角色的 ARN，格式为 `arn:aws:iam::account-id:role/role-name`。例如：`arn:aws:iam::111222333444:role/my-sns-access-role`。

Note

在 Systems Manager 控制台中，可在 Register run command task 页面上的 IAM Role 列表中选择此 ARN。有关信息，请参阅 [将任务分配给 Maintenance Window \(控](#)

制台) (p. 257)。在 Systems Manager API 中，在 `SendCommand` 请求中作为 `ServiceRoleArn` 的值输入此 ARN。

4. 选择查看策略。
5. 在 Name 框中键入名称，将其标识为允许发送 Amazon SNS 通知的策略。
14. 选择 Create policy。

任务 2：将 IAM PassRole 策略分配到 IAM 用户或组

When you register a task with a Maintenance Window, you specify a custom service role to run the actual task operations. This is the role that the service will assume when it runs tasks on your behalf. Before that, to register the task itself, you must assign the IAM `PassRole` policy to an IAM user account or an IAM group. This allows the IAM user or IAM group to specify, as part of registering those tasks with the Maintenance Window, the role that should be used when running tasks.

根据您将 `iam:Passrole` 权限分配给单独用户还是组，使用以下过程之一，提供将任务注册到 Maintenance Window 所需的最低权限。

将 IAM PassRole 策略分配给 IAM 用户账户

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 选择用户，然后选择您要更新的用户账户的名称。
3. 在权限选项卡的策略列表中，验证是否已列出 `AmazonSSMFullAccess` 策略，或是否存在可向 IAM 用户授予调用 Systems Manager API 的权限的类似策略。如果尚未包括，则添加权限。有关信息，请参阅 IAM User Guide 中的 [附加 IAM 策略 \(控制台\)](#)。
4. 选择添加内联策略。
5. 在创建策略页面的 Visual editor (可视编辑器) 选项卡中，在选择服务区域中选择 IAM。
6. 在操作区域中，选择 `PassRole`。

Tip

在筛选框中键入 `passr` 可快速找到 `PassRole`。

7. 选择资源行，然后选择添加 ARN。
8. 在 Specify ARN for role 字段中，粘贴您在上一过程中创建的角色 ARN，然后选择保存更改。
9. 选择查看策略。
10. 在查看策略页面的名称框中，键入名称以标识此 `PassRole` 策略，然后选择创建策略。

将 IAM PassRole 策略分配给 IAM 组

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中，选择 Groups。
3. 在组列表中，选择要将 `iam:PassRole` 权限分配到的组的名称。
4. 在内联策略区域中，执行下列操作之一：
 - 如果尚未添加任何内联策略，请选择单击此处。
 - 如果已添加了一个或多个内联策略，请选择创建组策略。
5. 选择策略生成器，然后选择选择。
6. 做出以下选择：
 1. 效果：允许
 2. AWS 服务：Identity and Access Management

3. 操作 : PassRole
4. Amazon 资源名称 (ARN) : 输入您在[任务 1 : 为 Maintenance Window 创建自定义服务角色 \(p. 247\)](#)中创建的Maintenance Window的 ARN
7. 选择 Add Statement , 然后选择 Next Step。
8. 选择 Apply Policy。

控制对 Maintenance Window 的访问 (AWS CLI)

以下过程介绍如何使用 AWS CLI 来创建Maintenance Window所需的角色和权限。

主题

- [任务 1 : 为 Maintenance Window 创建自定义服务角色 \(p. 250\)](#)
- [任务 2 : 将 IAM PassRole 策略分配到 IAM 用户或组 \(p. 251\)](#)

任务 1 : 为 Maintenance Window 创建自定义服务角色

1. 将以下信任策略复制并粘贴到文本文件中。使用以下名称和文件扩展名保存此文件 : mw-role-trust-policy.json。

Note

仅当要使用 Amazon SNS 发送与通过 [SendCommand](#) API 或 AWS CLI 中的 send-command 运行的Maintenance Window任务相关的通知时，才需要 "sns.amazonaws.com"。

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

2. 打开 AWS CLI 并在放置 mw-role-trust-policy.json 的目录中运行以下命令，创建一个名为 mw-task-role 的Maintenance Window角色。此命令将上一步中创建的策略分配给该角色。

```
aws iam create-role --role-name mw-task-role --assume-role-policy-document file://mw-role-trust-policy.json
```

系统将返回类似于以下内容的信息。

```
{
  "Role": {
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Effect": "Allow",
          "Principal": {
```

```
        "Service": [
            "ssm.amazonaws.com",
            "ec2.amazonaws.com",
            "sns.amazonaws.com"
        ]
    },
    "RoleId": "AROAIIZKPBKS2LEXAMPLE",
    "CreateDate": "2017-04-04T03:40:17.373Z",
    "RoleName": "mw-task-role",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/mw-task-role"
}
```

Note

记录 RoleName 和 Arn。在您创建 Maintenance Window 时，将指定这些内容。

3. 运行以下命令，将 AmazonSSMMaintenanceWindowRole 管理的策略附加到在步骤 2 中创建的角色。

```
aws iam attach-role-policy --role-name mw-task-role --policy-arn
arn:aws:iam::aws:policy/service-role/AmazonSSMMaintenanceWindowRole
```

任务 2：将 IAM PassRole 策略分配到 IAM 用户或组

When you register a task with a Maintenance Window, you specify a custom service role to run the actual task operations. This is the role that the service will assume when it runs tasks on your behalf. Before that, to register the task itself, you must assign the IAM [PassRole](#) policy to an IAM user account or an IAM group. This allows the IAM user or IAM group to specify, as part of registering those tasks with the Maintenance Window, the role that should be used when running tasks.

将 IAM PassRole 策略分配给 IAM 用户账户或组

1. 将以下 IAM 策略复制并粘贴到文本编辑器中，然后使用以下名称和文件扩展名保存：mw-passrole-policy.json。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1491345526000",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole",
        "ssm:RegisterTaskWithMaintenanceWindow"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

2. 打开 AWS CLI。
3. 根据您是否要将权限分配到 IAM 用户或组，运行以下任一命令。

- 对于 IAM 用户：

```
aws iam put-user-policy --user-name user-name --policy-name "policy-name" --policy-document path-to-document
```

对于 *user-name*，指定将向Maintenance Window分配任务的 IAM 用户。对于 *policy-name*，指定您用于标识策略的名称。对于 *path-to-document*，请指定步骤 1 中保存文件的路径。例如：file:///C:\Temp\mw-passrole-policy.json

Note

如果您计划使用 AWS Systems Manager 控制台注册Maintenance Window的任务，则还必须将 AmazonSSMFullAccess 策略分配给您的用户账户。运行以下命令将此策略分配给您的账户。

```
aws iam attach-user-policy --policy-arn arn:aws:iam::aws:policy/  
AmazonSSMFullAccess --user-name user-name
```

- 对于 IAM 组：

```
aws iam put-group-policy --group-name group-name --policy-name "policy-name" --  
policy-document path-to-document
```

对于 *group-name*，请指定其成员将任务分配到Maintenance Window的 IAM 组。对于 *policy-name*，指定要用于标识策略的名称。对于 *path-to-document*，请指定步骤 1 中保存文件的路径。例如：file:///C:\Temp\mw-passrole-policy.json

Note

如果您计划使用 AWS Systems Manager 控制台注册Maintenance Window的任务，则还必须将 AmazonSSMFullAccess 策略分配给您的组。运行以下命令将此策略分配给您的组。

```
aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/  
AmazonSSMFullAccess --group-name group-name
```

4. 运行以下命令验证策略是否已分配给该组。

```
aws iam list-group-policies --group-name group-name
```

控制对 Maintenance Window 的访问 (Tools for Windows PowerShell)

以下过程介绍如何使用 Tools for Windows PowerShell 来创建Maintenance Window所需的角色和权限。

主题

- [任务 1：为 Maintenance Window 创建自定义服务角色 \(p. 252\)](#)
- [任务 2：将 IAM PassRole 策略分配到 IAM 用户或组 \(p. 253\)](#)

任务 1：为 Maintenance Window 创建自定义服务角色

1. 将以下信任策略复制并粘贴到文本文件中。使用以下名称和文件扩展名保存此文件：mw-role-trust-policy.json。

Note

仅当要使用 Amazon SNS 发送与通过 [SendCommand](#) API 运行的 Maintenance Window 任务相关的通知时，才需要 "sns.amazonaws.com"。

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

2. 打开 Tools for Windows PowerShell 并运行以下命令来创建角色 (其名称将此角色标识为 Maintenance Window 角色)，例如 my-maintenance-window-role。该角色使用前一步中创建的策略。

```
New-IAMRole -RoleName "mw-task-role" -AssumeRolePolicyDocument (Get-Content -raw .\mw-
role-trust-policy.json)
```

系统将返回类似于以下内容的信息。

```
Arn : arn:aws:iam::123456789012:role/mw-task-role
AssumeRolePolicyDocument : ExampleDoc12345678
CreateDate : 4/4/2017 11:24:43
Path : /
RoleId : AROAIIZKPBKS2LEXAMPLE
RoleName : mw-task-role
```

3. 运行以下命令将 AmazonSSMMaintenanceWindowRole 管理的策略附加到上一步中创建的角色。

```
Register-IAMRolePolicy -RoleName mw-task-role -PolicyArn
arn:aws:iam::aws:policy/service-role/AmazonSSMMaintenanceWindowRole
```

任务 2：将 IAM PassRole 策略分配到 IAM 用户或组

When you register a task with a Maintenance Window, you specify a custom service role to run the actual task operations. This is the role that the service will assume when it runs tasks on your behalf. Before that, to register the task itself, you must assign the IAM [PassRole](#) policy to an IAM user account or an IAM group. This allows the IAM user or IAM group to specify, as part of registering those tasks with the Maintenance Window, the role that should be used when running tasks.

1. 将以下 IAM 策略复制并粘贴到文本编辑器中，并使用 .json 文件扩展名保存文件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1491345526000",
      "Effect": "Allow",
      "Action": [
```

```

        "iam:GetRole",
        "iam:PassRole",
        "ssm:RegisterTaskWithMaintenanceWindow"
    ],
    "Resource": [
        "*"
    ]
  }
]
}

```

2. 打开 Tools for Windows PowerShell。
3. 根据您是否要将权限分配到 IAM 用户或组，运行以下任一命令。

- 对于 IAM 用户：

```
Write-IAMUserPolicy -UserName user-name -PolicyDocument (Get-Content -raw path-to-document) -PolicyName policy-name
```

对于 *user-name*，指定将向Maintenance Window分配任务的 IAM 用户。对于 *policy-name*，指定您用于标识策略的名称。对于 *path-to-document*，请指定步骤 1 中保存文件的路径。例如：c:\temp\passrole-policy.json

Note

如果您计划使用 AWS Systems Manager 控制台注册Maintenance Window的任务，则还必须将 AmazonSSMFullAccess 策略分配给您的用户账户。运行以下命令将此策略分配给您的账户。

```
Register-IAMUserPolicy -UserName user-name -PolicyArn arn:aws:iam::aws:policy/AmazonSSMFullAccess
```

- 对于 IAM 组：

```
Write-IAMGroupPolicy -GroupName group-name -PolicyDocument (Get-Content -raw path-to-document) -PolicyName policy-name
```

对于 *group-name*，请指定将任务分配到Maintenance Window的 IAM 组。对于 *policy-name*，指定要用于标识策略的名称。对于 *path-to-document*，请指定步骤 1 中保存文件的路径。例如：c:\temp\passrole-policy.json

Note

如果您计划使用 AWS Systems Manager 控制台注册Maintenance Window的任务，则还必须将 AmazonSSMFullAccess 策略分配给您的用户账户。运行以下命令将此策略分配给您的组。

```
Register-IAMGroupPolicy -GroupName group-name -PolicyArn arn:aws:iam::aws:policy/AmazonSSMFullAccess
```

4. 运行以下命令验证策略是否已分配给该组。

```
Get-IAMGroupPolicies -GroupName group-name
```

排除 IAM Maintenance Window 权限故障

使用以下信息可帮助您解决有关 AWS Systems Manager 中 Maintenance Window 权限的常见问题。

编辑任务错误：在用于编辑 Maintenance Window 任务的页面上，IAM 角色列表会返回以下错误消息：“找不到为此任务指定的 IAM 维护时段角色。它可能已删除，也可能尚未创建。”

问题 1：您最初指定的 IAM Maintenance Window 角色在您创建任务后被删除。

可能的修复措施：选择其他 IAM Maintenance Window 角色（如果您的账户中存在），或者新建一个并为任务选择该角色。

问题 2：如果任务是使用 AWS CLI、Tools for Windows PowerShell 或 AWS SDK 创建的，则可能指定了不存在的 IAM Maintenance Window 角色名称。例如，IAM Maintenance Window 角色可能已在您创建任务之前删除，或者键入的角色名称不正确，如 **myrole**，而不是 **my-role**。

可能的修复措施：为您要使用的 IAM Maintenance Window 角色选择正确的名称，或者新建一个要为任务指定的角色。

使用Maintenance Window

本部分介绍如何创建、配置以及更新或删除Maintenance Window。本部分还介绍如何针对Maintenance Window的目标和任务执行这些相同的任务。

Important

我们建议您一开始在测试环境中创建和配置 Maintenance Window。

开始前的准备工作

在创建Maintenance Window之前，您必须配置对Maintenance Window的访问权限。有关更多信息，请参阅[控制对 Maintenance Window 的访问权限 \(p. 247\)](#)。

主题

- [创建 Maintenance Window \(控制台\) \(p. 255\)](#)
- [将目标分配给 Maintenance Window \(控制台\) \(p. 256\)](#)
- [将任务分配给 Maintenance Window \(控制台\) \(p. 257\)](#)
- [更新或删除Maintenance Window \(p. 257\)](#)

创建 Maintenance Window (控制台)

要创建Maintenance Window，您必须执行以下操作：

- 创建时段，并定义其计划和持续时间。
- 为时段分配目标。
- 分配要在时段内运行的任务。

当您完成这些步骤之后，Maintenance Window将按照您定义的计划运行，并对您指定的目标运行任务。每完成一项任务，Systems Manager 都会记录执行详情。

您可以对目标运行以下类型的任务：

- 通过使用 Systems Manager Run Command 的命令
- 使用 Systems Manager Automation 执行自动化工作流程
- 使用 AWS Lambda 执行函数
- 使用 AWS Step Functions 执行状态机

Note

AWS Systems Manager 控制台目前不支持运行 Step Functions。要注册此类型的任务，必须使用 AWS CLI。有关如何使用 AWS CLI 创建、配置和更新Maintenance Window的示例，请参阅 [Systems Manager Maintenance Window演练 \(p. 259\)](#)。

创建Maintenance Window

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Maintenance Window。
3. 选择 Create a Maintenance Window。
4. 在 Name 字段中，键入描述性名称，以帮助您将此Maintenance Window标识为测试Maintenance Window。
5. 在 Description 字段中输入描述。
6. 如果要允许Maintenance Window任务在托管实例上运行，那么即使没有将这些实例注册为目标，也应选择允许已注销目标。如果选择了此选项，您在将任务注册到Maintenance Window时就可以选择已注销实例 (按实例 ID)。

如果未选择此选项，您在将任务注册到Maintenance Window时必须选择之前注册的目标。

7. 使用其中一个计划选项为Maintenance Window指定计划。
8. 在 Duration 字段中，键入Maintenance Window应该运行的小时数。
9. 在 Stop initiating tasks 字段中，键入系统应该在Maintenance Window结束前几小时停止计划要运行的新任务。
10. 选择 Create maintenance window。系统将让您返回到Maintenance Window页面。您刚创建的 Maintenance Window的状态是 Enabled。

将目标分配给 Maintenance Window (控制台)

创建Maintenance Window后，分配将要运行任务的目标。

向Maintenance Window分配目标

1. 在Maintenance Window列表中，选择您刚创建的Maintenance Window。
2. 选择 Actions，然后选择 Register targets。
3. 在 Target Name 字段中，键入目标的名称。
4. 在 Description 字段中，键入描述。
5. 在所有者信息字段中，指定您的名字或工作别名。在此Maintenance Window中，对这些目标运行任务时引发的任何 CloudWatch Events 中都包含所有者信息。
6. 在 Select targets by 部分中，选择 Specifying Tags 以使用您之前分配给实例的 Amazon EC2 标签将实例设为目标。选择 Manually Selecting Instances 以根据各个实例的实例 ID 选择这些实例。

Note

如果您没有看到想设为目标的实例，请验证是否为 Systems Manager 配置了这些实例。有关更多信息，请参阅 [设置 AWS Systems Manager \(p. 6\)](#)。

7. 选择注册目标。

如果您想将更多目标分配到此时段，请选择 Targets 选项卡，然后选择 Register new targets。利用此选项，您可以选择不同的目标设定方式。例如，如果您之前按实例 ID 将实例设为目标，则可以注册新目标并通过指定 Amazon EC2 标签将实例设为目标。

将任务分配给 Maintenance Window (控制台)

分配目标后，您需分配在时段中执行的任务。

向 Maintenance Window 分配任务

1. 在 Maintenance Window 列表中，选择您刚创建的 Maintenance Window。
2. 选择操作，然后选择注册运行命令任务以使用 SSM 文档对目标运行您选择的命令，或选择注册自动化任务以使用 SSM Automation 文档对目标运行您选择的自动化工作流程。有关如何使用 AWS CLI 创建 Lambda 和 Step Functions 任务的示例，请参阅 [Systems Manager Maintenance Window 演练 \(p. 259\)](#)。
3. 在 Name 字段中，键入任务名称。
4. 在 Description 字段中，键入描述。
5. 从 Document 列表中，选择定义要运行的任务的 SSM Command 或 Automation 文档。
6. 在文档版本列表 (针对 Automation 任务) 中，选择要使用的文档版本。
7. 在任务优先级列表中，指定此任务的优先级。1 表示最高优先级。Maintenance Window 中的任务按优先级顺序计划，具有相同优先级的任务则并行计划。
8. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅 [我的实例在哪里？ \(p. 202\)](#) 中的故障排除提示。

9. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
10. 在 IAM 角色字段中，指定 Maintenance Window ARN。有关创建 Maintenance Window ARN 的更多信息，请参阅 [控制对 Maintenance Window 的访问权限 \(p. 247\)](#)。
 11. 在输入参数部分中，为文档指定参数。对于 Automation 文档，系统将自动填充某些值。您可以保留或更换这些值。
 12. 完成向导。

更新或删除 Maintenance Window

您可以更新或删除 Maintenance Window，也可以更新或删除 Maintenance Window 的目标或任务。如果要编辑 Maintenance Window 的详细信息，您可以更改计划、目标和任务。您还可以指定时段、目标和任务的名称和描述，从而可以更好地理解它们的用途和更轻松地管理您的时段队列。

本部分介绍如何使用 AWS Systems Manager 控制台来更新或删除 Maintenance Window、目标和任务。有关如何使用 AWS CLI 执行此操作的示例，请参阅 [演练：更新 Maintenance Window \(p. 267\)](#)。

更新或删除 Maintenance Window (控制台)

您可以更新某个 Maintenance Window，以更改名称、描述、时段计划以及该时段是否应允许已注销的目标。

更新或删除 Maintenance Window

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Maintenance Window。
3. 选择要更新或删除的 Maintenance Window，然后执行以下操作之一：
 - 选择 Delete。系统将提示您确认操作。
 - 选择 Edit。在编辑维护窗口页面中，更改所需的值和选项，然后选择编辑维护窗口。

更新或删除Maintenance Window的目标 (控制台)

您可以更新或删除Maintenance Window的目标。如果您选择更新Maintenance Window目标，则可以指定新的目标名称、描述和所有者。您也可以选择不同的目标。

更新或删除 Maintenance Window 的目标

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Maintenance Window。
3. 选择要更新的 Maintenance Window 的名称，然后执行以下操作之一：
 - 要更新目标，选择编辑。
 - 要删除目标，选择“取消注册目标”，然后选择目标选项卡。

选择要删除的目标，然后选择取消注册目标。如果您希望系统在删除某个目标之前检查该目标是否被任何任务引用，请在取消注册维护时段目标窗口中，将安全取消注册目标选项保持为选中状态。如果该目标被某个任务引用，系统将返回错误，但不会删除该目标。如果您希望系统即使在目标被某个任务引用的情况下仍然将其删除，请清除安全取消注册目标选项。

选择取消注册。

更新或删除Maintenance Window的任务

您可以更新或删除Maintenance Window的任务。如果您选择更新，则可以指定新的任务名称、描述和所有者。对于 Run Command 和 Automation 任务，可以为任务选择另一个 SSM 文档。但是，您无法编辑任务以更改其类型。例如，如果您创建了一个 Automation 任务，则无法编辑该任务并将其更改为 Run Command 任务。

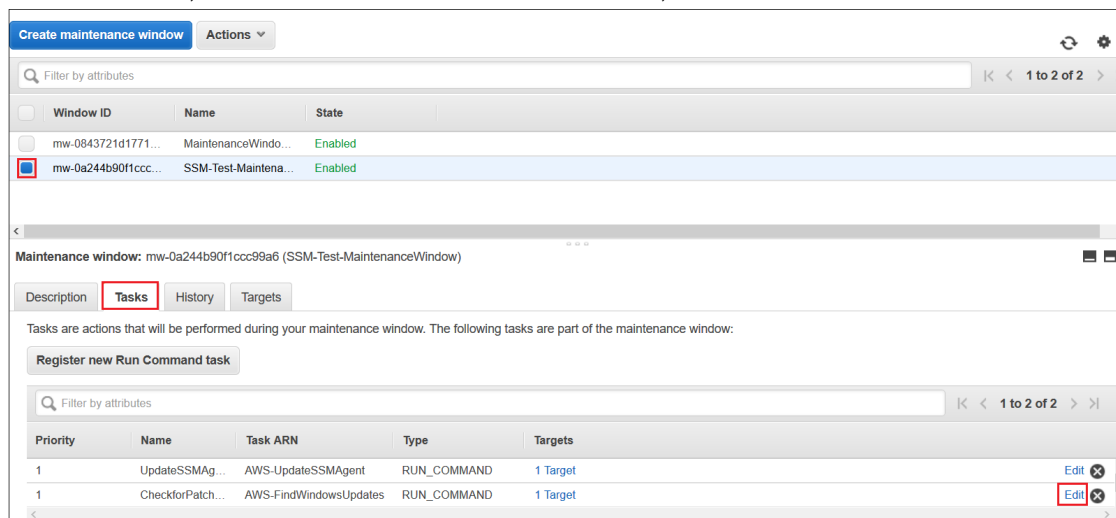
Note

以下过程描述了您在 Amazon EC2 控制中执行的步骤。您也可以在新的 [AWS Systems Manager 控制台](#) 中执行这些步骤。新控制台中的步骤将不同于下面的步骤。

更新或删除Maintenance Window的任务

1. 打开 [Amazon EC2 控制台](#)，展开导航窗格中的 Systems Manager Shared Resources，然后选择 Maintenance Windows。
2. 选择要更新的Maintenance Window。
3. 选择 Tasks 选项卡。

4. 如果要删除任务，请选择 Edit 旁边的小 X。如果要编辑任务，请选择 Edit。



5. 更改所需的值和选项，然后选择 Edit Task。系统 will 让您返回到 Maintenance Window 页面。

Systems Manager Maintenance Window演练

使用以下演练，借助 AWS CLI 来创建、配置和更新 Maintenance Window。在尝试这些演练之前，必须先配置 Maintenance Window 角色和权限。有关更多信息，请参阅 [控制对 Maintenance Window 的访问权限 \(p. 247\)](#)。

主题

- [演练：创建和配置 Maintenance Window \(CLI\) \(p. 259\)](#)
- [演练：更新 Maintenance Window \(p. 267\)](#)
- [演练：列出有关 Maintenance Window 的信息 \(p. 271\)](#)

演练：创建和配置 Maintenance Window (CLI)

以下演练介绍如何使用 AWS CLI 来创建和配置 Maintenance Window、目标和任务。

使用 AWS CLI 创建和配置 Maintenance Window

1. 将 AWS CLI [下载](#)到本地计算机上。
2. 打开 AWS CLI，然后运行以下命令，以创建从每个星期二下午 4 点开始运行 4 个小时、停止 1 小时且允许无关联目标的 Maintenance Window。有关创建 `schedule` 参数的 cron 表达式的更多信息，请参阅 [参考：Systems Manager 的 Cron 和 Rate 表达式 \(p. 418\)](#)。

```
aws ssm create-maintenance-window --name "My-First-Maintenance-Window" --schedule  
"cron(0 16 ? * TUE *)" --duration 4 --cutoff 1 --allow-unassociated-targets
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowId": "mw-ab12cd34ef56gh78"  
}
```

3. 运行以下命令，列出您的 AWS 账户中所有的 Maintenance Window。

```
aws ssm describe-maintenance-windows
```

系统将返回类似于以下内容的信息。

```
{
  "WindowIdentities":[
    {
      "Duration":4,
      "Cutoff":1,
      "WindowId":"mw-ab12cd34ef56gh78",
      "Enabled":true,
      "Name":"My-First-Maintenance-Window"
    }
  ]
}
```

4. 运行以下命令，将实例注册为此Maintenance Window的目标。系统将返回Maintenance Window目标ID。在后面的步骤中，您将使用此ID为此Maintenance Window注册任务。

```
aws ssm register-target-with-maintenance-window --window-id "mw-ab12cd34ef56gh78" --
target "Key=InstanceIds,Values=ID" --owner-information "Single instance" --resource-
type "INSTANCE"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}
```

您可以使用以下命令注册多个实例。

```
aws ssm register-target-with-maintenance-window --window-id "mw-ab12cd34ef56gh78" --
targets "Key=InstanceIds,Values=ID 1,ID 2" --owner-information "Two instances in a
list" --resource-type "INSTANCE"
```

您还可以使用EC2标签注册实例。

```
aws ssm register-target-with-maintenance-window --window-id "mw-ab12cd34ef56gh78" --
targets "Key=tag:Environment,Values=Prod" "Key=Role,Values=Web" --owner-information
"Production Web Servers" --resource-type "INSTANCE"
```

5. 使用以下命令，显示Maintenance Window的目标。

```
aws ssm describe-maintenance-window-targets --window-id "mw-ab12cd34ef56gh78"
```

系统将返回类似于以下内容的信息。

```
{
  "Targets":[
    {
      "ResourceType":"INSTANCE",
      "OwnerInformation":"Single instance",
      "WindowId":"mw-ab12cd34ef56gh78",
      "Targets":[
        {
          "Values":[
```

```

        "i-11aa22bb33cc44dd5"
      ],
      "Key": "InstanceIds"
    }
  ],
  "WindowTargetId": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4"
},
{
  "ResourceType": "INSTANCE",
  "OwnerInformation": "Two instances in a list",
  "WindowId": "mw-ab12cd34ef56gh78",
  "Targets": [
    {
      "Values": [
        "i-1a2b3c4d5e6f7g8h9",
        "i-aa11bb22cc33dd44e "
      ],
      "Key": "InstanceIds"
    }
  ],
  "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
},
{
  "ResourceType": "INSTANCE",
  "OwnerInformation": "Production Web Servers",
  "WindowId": "mw-ab12cd34ef56gh78",
  "Targets": [
    {
      "Values": [
        "Prod"
      ],
      "Key": "tag:Environment"
    },
    {
      "Values": [
        "Web"
      ],
      "Key": "tag:Role"
    }
  ],
  "WindowTargetId": "1111aaa-2222-3333-4444-1111aaa "
}
]
}

```

6. 运行以下命令，为Maintenance Window注册任务。第一个示例中的任务使用 Systems Manager Run Command 借助 AWS-RunShellScript 文档运行 `df` 命令。您也可以指定使用 Systems Manager Automation、AWS Lambda 和 AWS Step Functions 的任务，如其他示例所示。注册任务时，可以指定以下参数：
 - **targets**：指定 Key=WindowTargetIds,Values=**IDS** 以指定已注册到Maintenance Window的目标，或指定 Key=InstanceIds,Values=**IDS** 以将不确定是否已注册到Maintenance Window的各个实例设为目标。
 - **task-arn**：任务在执行期间使用的资源。对于 RUN_COMMAND 和 AUTOMATION 任务类型，TaskArn 是 SSM 文档名称或 ARN。对于 LAMBDA 任务，TaskArn 是函数名称或 ARN。对于 STEP_FUNCTION 任务，TaskArn 是状态机 ARN。
 - **window-id**：目标Maintenance Window的 ID。
 - **task-type**：任务的类型。类型可以是以下项之一：RUN_COMMAND、AUTOMATION、LAMBDA 或 STEP_FUNCTION。
 - **task-invocation-parameters**：必需和可选参数。下一个列表中介绍了一些常见 task-invocation-parameters 参数。

- **max-concurrency** : (可选) 允许同时运行命令的最大实例数。您可以指定一个数字或一个百分比 (如 10 或 10%)。
- **max-errors** : (可选) 允许的不会导致命令失败的最大错误数。当命令失败的次数超出 `MaxErrors` 值一次, 系统就会停止向其他目标发送命令。您可以指定一个数字或一个百分比 (如 10 或 10%)。
- **priority** : Maintenance Window 中的任务的优先级。数字越小, 优先级越高 (例如, 1 表示最高优先级)。Maintenance Window 中的任务按优先级顺序计划。具有相同优先级的任务则并行计划。

适用于 task-invocation-parameters 的常见参数

以下列表介绍了使用 task-invocation-parameters 时可以指定的一些常见参数。您可使用 `{{ PARAMETER_NAME }}` 语法指定这些参数, 如本部分中的示例所示。

- **TARGET_ID** : 目标的 ID。如果目标类型为 INSTANCE (目前唯一支持的类型), 则目标 ID 为实例 ID。
- **TARGET_TYPE** : 目标的类型。目前仅支持 INSTANCE。
- **WINDOW_ID** : 目标 Maintenance Window 的 ID。
- **WINDOW_TASK_ID** : 正在执行的时段任务的 ID。
- **WINDOW_TARGET_ID** : 包含目标的时段目标的 ID (目标 ID)。
- **LOGGING_S3_BUCKET_NAME** : Amazon S3 存储桶名称 (如果是使用 logging-info 参数配置的)。
- **LOGGING_S3_KEY_PREFIX** : Amazon S3 密钥前缀 (如果是使用 logging-info 参数配置的)。
- **LOGGING_S3_REGION** : Amazon S3 区域 (如果是使用 logging-info 参数配置的)。
- **WINDOW_EXECUTION_ID** : 当前时段执行的 ID。
- **TASK_EXECUTION_ID** : 当前任务执行的 ID。
- **INVOCATION_ID** : 当前调用的 ID。

```
aws ssm register-task-with-maintenance-window --window-id mw-ab12cd34ef56gh78 --task-arn "AWS-RunShellScript" --targets "Key=InstanceIds,Values=Instance ID" --service-role-arn "arn:aws:iam::1122334455:role/MW-Role" --task-type "RUN_COMMAND" --task-invocation-parameters '{"RunCommand":{"Parameters":{"commands":["df"]}}}' --max-concurrency 1 --max-errors 1 --priority 10
```

系统将返回类似于以下内容的信息。

```
{
  "WindowTaskId": "44444444-5555-6666-7777-88888888"
}
```

您还可以使用 Maintenance Window 目标 ID 注册任务。Maintenance Window 目标 ID 通过之前的命令返回。

```
aws ssm register-task-with-maintenance-window --targets "Key=WindowTargetIds,Values=Window Target ID" --task-arn "AWS-RunShellScript" --service-role-arn "arn:aws:iam::1122334455:role/MW-Role" --window-id "mw-ab12cd34ef56gh78" --task-type "RUN_COMMAND" --task-invocation-parameters '{"RunCommand":{"Parameters":{"commands":["df"]}}}' --max-concurrency 1 --max-errors 1 --priority 10
```

系统将返回类似于以下内容的信息。

```
{
  "WindowTaskId": "44444444-5555-6666-7777-88888888"
}
```


}

以下示例演示如何注册其他任务类型。

Important

Maintenance Window 的 IAM 策略要求您使用 SSM 作为 Lambda 函数 (或别名) 名称和 Step Functions 状态机名称的前缀，如下面的前两个示例所示。在继续注册这些类型的任务之前，必须在 AWS Lambda 和 AWS Step Functions 中将它们的名称更新为包含 SSM。

Lambda

```
aws ssm register-task-with-maintenance-window --window-id "mw-0290d787d641f11f3"
--targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa
--task-arn arn:aws:lambda:us-east-2:711106535523:function:SSMTestFunction
--service-role-arn arn:aws:iam::711106535523:role/MaintenanceWindows
--task-type LAMBDA --task-invocation-parameters '{"Lambda":
{"Payload": "{\\"targetId\\":\\"{{TARGET_ID}}\\",\\"targetType\\":
\\"{{TARGET_TYPE}}\\"}",\\"Qualifier\\":\\"$LATEST\\",\\"ClientContext\\":\\"ew0KICAiY3VzdG9tIjogew0KICAgICJjbGllbn"}'
--priority 0 --max-concurrency 10 --max-errors 5 --name "Lambda_Example" --description
"My Lambda Example"
```

Step Functions

```
aws ssm register-task-with-maintenance-window --window-id "mw-0290d787d641f11f3"
--targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --
task-arn arn:aws:states:us-east-2:711106535523:stateMachine:SSMTestStateMachine
--service-role-arn arn:aws:iam::711106535523:role/MaintenanceWindows --task-type
STEP_FUNCTIONS --task-invocation-parameters '{"StepFunctions":{"Input":{"instanceId
\\":\\"{{TARGET_ID}}\\"}}}' --priority 0 --max-concurrency 10 --max-errors 5 --name
"Step_Functions_Example" --description "My Step Functions Example"
```

自动化

```
aws ssm register-task-with-maintenance-window --window-id "mw-0290d787d641f11f3"
--targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --
task-arn AutomationDocumentName --service-role-arn arn:aws:iam::711106535523:role/
MaintenanceWindows --task-type AUTOMATION --task-invocation-parameters
"Automation={DocumentVersion=5,Parameters={instanceId='{{TARGET_ID}}'}}" --priority 0
--max-concurrency 10 --max-errors 5 --name Name --description Description
```

Run Command

```
aws ssm register-task-with-maintenance-window --window-id "mw-0290d787d641f11f3"
--targets Key=WindowTargetIds,Values=31547414-69c3-49f8-95b8-ed2dcf045faa --task-
arn AWS-RunPowerShellScript --service-role-arn arn:aws:iam::711106535523:role/
MaintenanceWindows --task-type RUN_COMMAND --task-invocation-parameters
"RunCommand={Comment=SomeComment,DocumentHashType=Sha256,DocumentHash=b9d0966408047ebcafee82de4d42
west-2:711106535523:RunCommandTopic,NotificationEvents=[Success,Failed],NotificationType=Invocation
RunCommand,TimeoutSeconds=30,Parameters={commands=ipconfig}}" --priority 0 --max-
concurrency 10 --max-errors 5 --name "Run_Command_Sample" --description "My Run Command
Sample"
```

- 运行以下命令，列出Maintenance Window已注册的所有任务。

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78"
```

系统将返回类似于以下内容的信息。


```
{
  "Tasks":[
    {
      "ServiceRoleArn":"arn:aws:iam::111111111:role/MW-Role",
      "MaxErrors":"1",
      "TaskArn":"AWS-RunPowerShellScript",
      "MaxConcurrency":"1",
      "WindowTaskId":"3333-3333-3333-333333",
      "TaskParameters":{"
        "commands":{"
          "Values":[
            "driverquery.exe"
          ]
        }
      },
      "Priority":3,
      "Type":"RUN_COMMAND",
      "Targets":[
        {
          "Values":[
            "i-1a2b3c4d5e6f7g8h9"
          ],
          "Key":"InstanceIds"
        }
      ]
    },
    {
      "ServiceRoleArn":"arn:aws:iam::222222222:role/MW-Role",
      "MaxErrors":"1",
      "TaskArn":"AWS-RunPowerShellScript",
      "MaxConcurrency":"1",
      "WindowTaskId":"44444-44-44-444444",
      "TaskParameters":{"
        "commands":{"
          "Values":[
            "ipconfig.exe"
          ]
        }
      },
      "Priority":1,
      "Type":"RUN_COMMAND",
      "Targets":[
        {
          "Values":[
            "5555555-555555-555-5555555"
          ],
          "Key":"WindowTargetIds"
        }
      ]
    }
  ]
}
```

8. 运行以下命令，查看特定Maintenance Window的任务执行列表。

```
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowExecutions":[
    {
```

```

        "Status": "SUCCESS",
        "WindowExecutionId": "1111-1111-1111-1111",
        "StartTime": 1478230495.469
    },
    {
        "Status": "SUCCESS",
        "WindowExecutionId": "2222-2-2-22222222-22",
        "StartTime": 1478231395.677
    },
    # ... omitting a number of entries in the interest of space...
    {
        "Status": "SUCCESS",
        "WindowExecutionId": "33333-333-333-33333333",
        "StartTime": 1478272795.021
    },
    {
        "Status": "SUCCESS",
        "WindowExecutionId": "4444-44-44-44444444",
        "StartTime": 1478273694.932
    }
],
"NextToken": "111111 ... "
}

```

9. 运行以下命令，获取有关Maintenance Window任务执行的信息。

```

aws ssm get-maintenance-window-execution --window-execution-id
"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"

```

系统将返回类似于以下内容的信息。

```

{
  "Status": "SUCCESS",
  "TaskIds": [
    "333-33-3333-333333"
  ],
  "StartTime": 1478230495.472,
  "EndTime": 1478230516.505,
  "WindowExecutionId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
}

```

10. 运行以下命令，列出Maintenance Window执行过程中运行的任务。

```

aws ssm describe-maintenance-window-execution-tasks --window-execution-id
"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"

```

系统将返回类似于以下内容的信息。

```

{
  "WindowExecutionTaskIdentities": [
    {
      "Status": "SUCCESS",
      "EndTime": 1478230516.425,
      "StartTime": 1478230495.782,
      "TaskId": "33333-333-333-33333333"
    }
  ]
}

```

11. 运行以下命令，获取有关任务执行的详情。

```
aws ssm get-maintenance-window-execution-task --window-execution-id
"555555-555-55-555555" --task-id "4444-4444-4444-444444"
```

系统将返回类似于以下内容的信息。

```
{
  "Status": "SUCCESS",
  "MaxErrors": "1",
  "TaskArn": "AWS-RunPowerShellScript",
  "MaxConcurrency": "1",
  "ServiceRole": "arn:aws:iam::3333333333:role/MW-Role",
  "WindowExecutionId": "555555-555-55-555555",
  "Priority": 0,
  "StartTime": 1478230495.782,
  "EndTime": 1478230516.425,
  "Type": "RUN_COMMAND",
  "TaskParameters": [
  ],
  "TaskExecutionId": "4444-4444-4444-444444"
}
```

12. 运行以下命令，获取执行某个任务时执行的具体任务调用。

```
aws ssm describe-maintenance-window-execution-task-invocations --window-execution-id
"555555-555-55-555555" --task-id "4444-4444-4444-444444"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowExecutionTaskInvocationIdentities": [
    {
      "Status": "SUCCESS",
      "Parameters": "{ \" documentName \" : \" AWS-RunPowerShellScript \" , \"
instanceIds \" : [ \" i-1a2b3c4d5e6f7g8h9 \" , \" i-0a
00def7faa94fldc \" ], \" parameters \" : { \" commands \" : [ \" ipconfig.exe \" ]}, \"
maxConcurrency \" : \" 1 \" , \" maxErrors \" : \" 1 \" }",
      "ExecutionId": "555555-555-55-555555",
      "InvocationId": "3333-33333-3333-33333",
      "StartTime": 1478230495.842,
      "EndTime": 1478230516.291
    }
  ]
}
```

13. 如果需要，请运行以下命令以删除您创建的Maintenance Window。

```
aws ssm delete-maintenance-window --window-id "mw-1a2b3c4d5e6f7g8h9"
```

系统将返回类似于以下内容的信息：

```
{
  "WindowId": "mw-1a2b3c4d5e6f7g8h9"
}
```

演练：更新 Maintenance Window

本部分介绍如何使用 AWS CLI 更新 Maintenance Window。本部分还包含有关更新不同的任务类型 (包括 Systems Manager Run Command、Systems Manager Automation、AWS Lambda 和 AWS Step Functions 任务) 的信息。有关更新 Maintenance Window 的更多信息，请参阅[更新或删除 Maintenance Window \(p. 257\)](#)。

本部分中的示例使用以下 Systems Manager 操作来更新 Maintenance Window。

- [UpdateMaintenanceWindow](#)
- [UpdateMaintenanceWindowTarget](#)
- [UpdateMaintenanceWindowTask](#)
- [DeregisterTargetFromMaintenanceWindow](#)

更新 Maintenance Window

1. 运行以下命令，将目标更新为包括名称和描述。

```
aws ssm update-maintenance-window-target --window-id "mw-12345678910" --window-target-id "a1b2c3d4-e5f6-g7h8i9" --name "NewTargetName" --description "NewTargetName description"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowId": "mw-12345678910",
  "WindowTargetId": "a1b2c3d4-e5f6-g7h8i9",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "i-aabbccddeeff"
      ]
    }
  ],
  "Name": "NewTargetName",
  "Description": "NewTargetName description"
}
```

2. 运行以下命令，使用 `replace` 选项来删除描述字段和添加更多目标。描述字段被删除，因为更新未包含该字段 (一个 null 值)。

```
aws ssm update-maintenance-window-target --window-id "mw-12345678910" --window-target-id "a1b2c3d4-e5f6-g7h8i9" --targets "Key=InstanceIds,Values=i-aabbccddeeff,i-223344556677" --name "NewTargetName" --replace
```

系统将返回类似于以下内容的信息。

```
{
  "WindowId": "mw-12345678910",
  "WindowTargetId": "a1b2c3d4-e5f6-g7h8i9",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "i-aabbccddeeff",
        "i-223344556677"
      ]
    }
  ]
}
```

```
    }
  ],
  "Name": "NewTargetName"
}
```

3. 运行以下命令可更新 Run Command 任务。

```
aws ssm update-maintenance-window-task --window-id "mw-12345678910" --window-
task-id "1111-2222-3333-4444-5555" --targets "Key=WindowTargetIds,Values=a1b2c3d4-
e5f6-g7h8i9c" --task-arn "AWS-RunPowerShellScript" --service-role-arn
"arn:aws:iam::abcdefghijkl:role/MaintenanceWindowsRole" --task-invocation-parameters
"RunCommand={Comment=A_Comment,Parameters={commands=ipconfig}}" --priority 1 --max-
concurrency 10 --max-errors 4 --name "RC_Name" --description "RC_Name description
extra"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowId": "mw-12345678916",
  "WindowTaskId": "aaa-bbb-ccc-ddd",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "a1b2c3d4-e5f6-g7h8i9c"
      ]
    }
  ],
  "TaskArn": "AWS-RunPowerShellScript",
  "ServiceRoleArn": "arn:aws:iam::abcdefghijkl:role/MaintenanceWindowsRole",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "RunCommand": {
      "Comment": "SomeComment",
      "Parameters": {
        "commands": [
          "ipconfig -tail"
        ]
      }
    }
  },
  "Priority": 1,
  "MaxConcurrency": "10",
  "MaxErrors": "4",
  "Name": "RC_Name",
  "Description": "RC_Name description extra"
}
```

4. 运行以下命令，向 Lambda 任务添加名称和描述。

```
aws ssm update-maintenance-window-task --window-id mw-1234567891 --window-
task-id 1a2b3c4d-5e6f-7g8h90 --targets "Key=WindowTargetIds,Values=a1b2c3d4-
e5f6-g7h8i9c,4444-555555-66666-7777" --task-arn "arn:aws:lambda:us-
east-2:1313131313:function:SSMTestLambda" --service-role-arn
"arn:aws:iam::abcdefghijkl:role/MaintenanceWindowsRole" --task-invocation-
parameters '{"Lambda":{"Payload":"{\"targetId\":\"{{TARGET_ID}}\",\"targetType\"":
\"{{TARGET_TYPE}}\"}}' --priority 0 --max-concurrency 10 --max-errors 5 --name
"TestLambda_Name" --description "My Rename Test"
```

系统将返回类似于以下内容的信息。

```
{
```

```
{
  "WindowId": "mw-1234567891",
  "WindowTaskId": "1a2b3c4d-5e6f-7g8h90",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "a1b2c3d4-e5f6-g7h8i9c",
        "4444-555555-66666-7777d"
      ]
    }
  ],
  "TaskArn": "arn:aws:lambda:us-east-2:1313131313:function:SSMTestLambda",
  "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindowsRole",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "Lambda": {
      "Payload": "e30="
    }
  },
  "Priority": 0,
  "MaxConcurrency": "10",
  "MaxErrors": "5",
  "Name": "TestLambda_Name",
  "Description": "TestLambda_Name description"
}
```

5. 运行以下命令，更新 AWS Step Functions 任务以更新 task-invocation-parameters。

```
aws ssm update-maintenance-window-task --window-id "mw-1234567891" --window-task-id
"1a2b3c4d-5e6f-7g8h9i" --targets "Key=WindowTargetIds,Values=a1b2c3d4-e5f6-g7h8i9c" --
task-arn "arn:aws:states:us-east-2:4242424242:execution:SSMStepFunctionTest" --service-
role-arn "arn:aws:iam::abcdefghijk:role/MaintenanceWindowsRole" --task-invocation-
parameters '{"StepFunctions":{"Input":{"instanceId":"{{ TARGET_ID }}\\"}}}' --
priority 0 --max-concurrency 10 --max-errors 5 --name "Update_Parameters" --description
"Test to update task invocation parameters"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowId": "mw-1234567891",
  "WindowTaskId": "1a2b3c4d-5e6f-7g8h9i",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "a1b2c3d4-e5f6-g7h8i9c"
      ]
    }
  ],
  "TaskArn": "arn:aws:states:us-east-2:4242424242:execution:SSMStepFunctionTest",
  "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindowsRole",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "StepFunctions": {
      "Input": "{\\"instanceId\\":\\"{{ TARGET_ID }}\\"}"
    }
  },
  "Priority": 0,
  "MaxConcurrency": "10",
  "MaxErrors": "5",
  "Name": "TestStepFunction_Task",
  "Description": "TestStepFunction_Task description"
}
```

6. 运行以下命令，从Maintenance Window中注销目标。本示例使用 `safe` 参数确定目标是否被任何任务引用，从而确定注销是否安全。

```
aws ssm deregister-target-from-maintenance-window --window-id "mw-1234567891b" --window-target-id "aaaa-bbbb-cccc-dddd" --safe
```

系统将返回类似于以下内容的信息。

```
An error occurred (TargetInUseException) when calling the
DeregisterTargetFromMaintenanceWindow operation: This Target cannot be deregistered
because it is still referenced in Task: a11b22c33d44e55f66
```

7. 运行以下命令可从Maintenance Window中注销目标，即使该目标已被某个任务引用。您可以使用 `no-safe` 参数强制执行注销操作。

```
aws ssm deregister-target-from-maintenance-window --window-id "mw-1234567891b" --window-target-id "aaaa-bbbb-cccc-dddd" --no-safe
```

系统将返回类似于以下内容的信息。

```
{
  "WindowId": "mw-1234567891b",
  "WindowTargetId": "aaaa-bbbb-cccc-ddd"
}
```

8. 运行以下命令可更新 Run Command 任务。本示例使用名为 `UpdateLevel` 的 Systems Manager Parameter Store 参数，其格式如下：`{{ssm:UpdateLevel}}`

```
aws ssm update-maintenance-window-task --window-id
"mw-1234567891b" --window-task-id "777-8888-9999-0000" --targets
"Key=InstanceIds,Values=i-yyyzzzzxxx111222" --task-invocation-parameters
"RunCommand={Comment=SomeComments,Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowId": "mw-1234567891b",
  "WindowTaskId": "777-8888-9999-0000",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "i-yyyzzzzxxx1112223"
      ]
    }
  ],
  "TaskArn": "AWS-InstallMissingWindowsUpdates",
  "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindows",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "RunCommand": {
      "Comment": "SomeComments",
      "Parameters": {
        "UpdateLevel": [
          "{{ssm:UpdateLevel}}"
        ]
      }
    }
  }
},
```

```

    "Priority": 0,
    "MaxConcurrency": "10",
    "MaxErrors": "5",
    "Name": "TracyMWTest_RunCommand2",
    "Description": "Test_RunCommandandParameterStore description"
  }

```

- 运行以下命令可更新 Automation 任务，为 task-invocation-parameters 参数指定 WINDOW_ID 和 WINDOW_TASK_ID 参数。

```

aws ssm update-maintenance-window-task --window-id "mw-1234567891b" --window-
task-id "777-8888-9999-000" --targets "Key=WindowTargetIds,Values=999-aaa-888-
bbb-777 --task-arn "AutoTestDoc" --service-role-arn arn:aws:iam::801422537783:role/
MaintenanceWindowsRoleTesting --task-invocation-parameters
  "Automation={Parameters={instanceId='{{TARGET_ID}}',initiator='{{WINDOW_ID}}.Task-
{{WINDOW_TASK_ID}}'}}" --priority 0 --max-concurrency 10 --max-errors 5

```

系统将返回类似于以下内容的信息。

```

{
  "WindowId": "mw-0a097ccb2abd5775b",
  "WindowTaskId": "777-8888-9999-0000",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "999-aaa-888-bbb-777"
      ]
    }
  ],
  "TaskArn": "AutoTestDoc",
  "ServiceRoleArn": "arn:aws:iam::abcdefghijk:role/MaintenanceWindows",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "Automation": {
      "Parameters": {
        "multi": [
          "{{WINDOW_TASK_ID}}"
        ],
        "single": [
          "{{WINDOW_ID}}"
        ]
      }
    }
  },
  "Priority": 0,
  "MaxConcurrency": "10",
  "MaxErrors": "5",
  "Name": "TestAutomation_Task",
  "Description": "TestAutomation_Task description"
}

```

演练：列出有关 Maintenance Window 的信息

本部分包含的命令可帮助您更新 Maintenance Window、任务、执行和调用或获取相关信息。

列出您的 AWS 账户中所有的 Maintenance Window

按如下所示运行命令。


```
aws ssm describe-maintenance-windows
```

系统将返回类似于以下内容的信息。

```
{
  "WindowIdentities":[
    {
      "Duration":2,
      "Cutoff":0,
      "WindowId":"mw-ab12cd34ef56gh78",
      "Enabled":true,
      "Name":"IAD-Every-15-Minutes"
    },
    {
      "Duration":4,
      "Cutoff":1,
      "WindowId":"mw-1a2b3c4d5e6f7g8h9",
      "Enabled":true,
      "Name":"My-First-Maintenance-Window"
    },
    {
      "Duration":8,
      "Cutoff":2,
      "WindowId":"mw-123abc456def789",
      "Enabled":false,
      "Name":"Every-Day"
    }
  ]
}
```

列出所有已启用的Maintenance Window

按如下所示运行命令。

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=true"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowIdentities":[
    {
      "Duration":2,
      "Cutoff":0,
      "WindowId":"mw-ab12cd34ef56gh78",
      "Enabled":true,
      "Name":"IAD-Every-15-Minutes"
    },
    {
      "Duration":4,
      "Cutoff":1,
      "WindowId":"mw-1a2b3c4d5e6f7g8h9",
      "Enabled":true,
      "Name":"My-First-Maintenance-Window"
    }
  ]
}
```

列出所有已禁用的Maintenance Window

按如下所示运行命令。

```
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=false"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowIdentities":[
    {
      "Duration":8,
      "Cutoff":2,
      "WindowId":"mw-1a2b3c4d5e6f7g8h9",
      "Enabled":false,
      "Name":"Every-Day"
    }
  ]
}
```

按名称筛选

在本示例中，命令将返回所有名称以“My”开头的Maintenance Window。

```
aws ssm describe-maintenance-windows --filters "Key=Name,Values=My"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowIdentities":[
    {
      "Duration":4,
      "Cutoff":1,
      "WindowId":"mw-1a2b3c4d5e6f7g8h9",
      "Enabled":true,
      "Name":"My-First-Maintenance-Window"
    }
  ]
}
```

显示匹配特定所有者信息值的Maintenance Window的目标

按如下所示运行命令。

```
aws ssm describe-maintenance-window-targets --window-id "mw-ab12cd34ef56gh78" --filters
"Key=OwnerInformation,Values=Single instance"
```

系统将返回类似于以下内容的信息。

```
{
  "Targets":[
    {
      "TargetType":"INSTANCE",
      "TagFilters":[
      ],
      "TargetIds":[
        "i-1a2b3c4d5e6f7g8h9"
      ],
      "WindowTargetId":"1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2",
      "OwnerInformation":"Single instance"
    }
  ]
}
```

```
}
```

显示将调用 AWS-RunPowerShellScript Run Command 的所有已注册任务

按如下所示运行命令。

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters  
"Key=TaskArn,Values=AWS-RunPowerShellScript"
```

系统将返回类似于以下内容的信息。

```
{  
  "Tasks": [  
    {  
      "ServiceRoleArn": "arn:aws:iam::444444444444:role/MW-Role",  
      "MaxErrors": "1",  
      "TaskArn": "AWS-RunPowerShellScript",  
      "MaxConcurrency": "1",  
      "WindowTaskId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e6c",  
      "TaskParameters": {  
        "commands": {  
          "Values": [  
            "driverquery.exe"  
          ]  
        }  
      },  
      "Priority": 3,  
      "Type": "RUN_COMMAND",  
      "Targets": [  
        {  
          "TaskTargetId": "i-1a2b3c4d5e6f7g8h9",  
          "TaskTargetType": "INSTANCE"  
        }  
      ]  
    },  
    {  
      "ServiceRoleArn": "arn:aws:iam::333333333333:role/MW-Role",  
      "MaxErrors": "1",  
      "TaskArn": "AWS-RunPowerShellScript",  
      "MaxConcurrency": "1",  
      "WindowTaskId": "33333-33333-333-33333",  
      "TaskParameters": {  
        "commands": {  
          "Values": [  
            "ipconfig.exe"  
          ]  
        }  
      },  
      "Priority": 1,  
      "Type": "RUN_COMMAND",  
      "Targets": [  
        {  
          "TaskTargetId": "44444-444-4444-4444444",  
          "TaskTargetType": "WINDOW_TARGET"  
        }  
      ]  
    }  
  ]  
}
```

显示优先级为 3 的所有已注册任务

按如下所示运行命令。

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters  
"Key=Priority,Values=3"
```

系统将返回类似于以下内容的信息。

```
{  
  "Tasks": [  
    {  
      "ServiceRoleArn": "arn:aws:iam::222222222:role/MW-Role",  
      "MaxErrors": "1",  
      "TaskArn": "AWS-RunPowerShellScript",  
      "MaxConcurrency": "1",  
      "WindowTaskId": "3333333-333-33333-33333",  
      "TaskParameters": {  
        "commands": {  
          "Values": [  
            "driverquery.exe"  
          ]  
        }  
      },  
      "Priority": 3,  
      "Type": "RUN_COMMAND",  
      "Targets": [  
        {  
          "TaskTargetId": "i-1a2b3c4d5e6f7g8h9",  
          "TaskTargetType": "INSTANCE"  
        }  
      ]  
    }  
  ]  
}
```

显示优先级为 1 且使用 Run Command 的所有已注册任务

按如下所示运行命令。

```
aws ssm describe-maintenance-window-tasks --window-id "mw-ab12cd34ef56gh78" --filters  
"Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

系统将返回类似于以下内容的信息。

```
{  
  "Tasks": [  
    {  
      "ServiceRoleArn": "arn:aws:iam::333333333:role/MW-Role",  
      "MaxErrors": "1",  
      "TaskArn": "AWS-RunPowerShellScript",  
      "MaxConcurrency": "1",  
      "WindowTaskId": "66666-555-66-555-6666",  
      "TaskParameters": {  
        "commands": {  
          "Values": [  
            "ipconfig.exe"  
          ]  
        }  
      },  
      "Priority": 1,  
      "Type": "RUN_COMMAND",  
      "Targets": [  
        {  
          "TaskTargetId": "777-77-777-7777777",  
          "TaskTargetType": "WINDOW_TARGET"  
        }  
      ]  
    }  
  ]  
}
```

```
}  
  ]  
}  
]  
}
```

列出在某个日期之前运行的所有任务

按如下所示运行命令。

```
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78" --filters  
"Key=ExecutedBefore,Values=2016-11-04T05:00:00Z"
```

系统将返回类似于以下内容的信息。

```
{  
  "WindowExecutions":[  
    {  
      "Status":"SUCCESS",  
      "EndTime":1478229594.666,  
      "WindowExecutionId": "",  
      "StartTime":1478229594.666  
    },  
    {  
      "Status":"SUCCESS",  
      "WindowExecutionId": "06dc5f8a-9ef0-4ae9-a466-ada2d4ce2d22",  
      "StartTime":1478230495.469  
    },  
    {  
      "Status":"SUCCESS",  
      "WindowExecutionId": "57ad6419-023e-44b0-a831-6687334390b2",  
      "StartTime":1478231395.677  
    },  
    {  
      "Status":"SUCCESS",  
      "WindowExecutionId": "ed1372b7-866b-4d64-bc2a-bbfd5195f4ae",  
      "StartTime":1478232295.529  
    },  
    {  
      "Status":"SUCCESS",  
      "WindowExecutionId": "154eb2fa-6390-4cb7-8c9e-55686b88c7b3",  
      "StartTime":1478233195.687  
    },  
    {  
      "Status":"SUCCESS",  
      "WindowExecutionId": "1c4de752-fff6-4778-b477-1681c6c03cf1",  
      "StartTime":1478234095.553  
    },  
    {  
      "Status":"SUCCESS",  
      "WindowExecutionId": "56062f75-e4d8-483f-b5c2-906d613409a4",  
      "StartTime":1478234995.12  
    }  
  ]  
}
```

列出在某个日期之后运行的所有任务

按如下所示运行命令。

```
aws ssm describe-maintenance-window-executions --window-id "mw-ab12cd34ef56gh78" --filters  
"Key=ExecutedAfter,Values=2016-11-04T17:00:00Z"
```

系统将返回类似于以下内容的信息。

```
{
  "WindowExecutions":[
    {
      "Status":"SUCCESS",
      "WindowExecutionId":"33333-4444-444-5555555",
      "StartTime":1478279095.042
    },
    {
      "Status":"SUCCESS",
      "WindowExecutionId":"55555-6666-6666-777777",
      "StartTime":1478279994.958
    },
    {
      "Status":"SUCCESS",
      "WindowExecutionId":"8888-888-888-888888",
      "StartTime":1478280895.149
    }
  ]
}
```

AWS Systems Manager 状态管理器

AWS Systems Manager 状态管理器 是一项安全、可扩展的配置管理服务，可以自动执行将您的 Amazon EC2 和混合基础设施保持在您定义的状态的进程。

借助 状态管理器，可以让某些任务按指定的计划自动运行，例如：

- 在启动时通过特定软件引导实例
- 按照既定计划下载和更新代理，包括 SSM 代理
- 配置网络设置
- 将实例加入 Windows 域 (仅限 Windows 实例)
- 在实例的整个生命周期内利用软件更新修补实例
- 在 Linux 和 Windows 托管的实例的整个生命周期内对这些实例运行脚本

状态管理器 可与 AWS CloudTrail 集成以提供您可审核的所有执行的记录，它还可与 Amazon CloudWatch Events 集成以跟踪状态更改。您也可以选择在 Amazon S3 中存储和查看详细的命令输出。

状态管理器 入门

要开始使用 状态管理器，请完成下表中描述的任务。

任务	了解更多信息
将托管实例上的 SSM 代理更新到最新版本	安装和配置 SSM 代理 (p. 14)
验证 Systems Manager 先决条件	Systems Manager 先决条件 (p. 6)
选择预定义的 AWS 命令或策略类型文档并在运行时指定参数 -或者- 创建定义要在实例上执行的操作的文档。	创建 Systems Manager 文档 (p. 297)
创建关联并将其应用到实例上	创建关联 (控制台) (p. 280)

相关内容

有关如何使用 状态管理器 的其他示例，请参阅以下博客文章：

- [使用 Amazon EC2 Systems Manager 和 Windows PowerShell DSC 来防止配置偏差](#)
- [使用 Amazon EC2 Systems Manager Run Command 和 状态管理器 运行 Ansible Playbook](#)
- [使用 状态管理器 在 Auto Scaling 组中配置 Amazon EC2 实例](#)

主题

- [关于 状态管理器 \(p. 278\)](#)
- [示例 状态管理器 文档 \(p. 279\)](#)
- [在 Systems Manager 中使用关联 \(p. 280\)](#)
- [Systems Manager 状态管理器 演练 \(p. 285\)](#)

关于 状态管理器

Systems Manager 状态管理器 是一项安全的、可扩展的配置管理服务，可确保您的 Amazon EC2 和混合基础设施处于您定义的所需状态或一致状态。

状态管理器 工作方式如下：

1. 确定要应用于托管实例的状态。

例如，确定要引导的应用程序或要配置的网络设置。在运行时，可使用 AWS 预配置文档指定状态的详细信息作为参数。您也可以创建自己的文档并直接在文档中指定状态，或在运行时指定状态作为参数。这些文档以 JSON 或 YAML 格式编写，称为 SSM 文档。

SSM 文档可包含多个操作或步骤（例如，要运行的多条命令）。状态管理器 使用的两种 SSM 文档类型为命令 文档和策略 文档。有关 SSM 文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

2. 您指定应用状态的时间或频率的计划。您可以指定 cron 或 rate 表达式。
3. 您为状态指定目标。

您可以将托管实例 (Amazon EC2 实例，或混合环境中为 Systems Manager 配置的计算机) 设为目标。您可以通过指定一个或多个实例 ID 来以实例为目标，也可以通过将 EC2 标签设为目标来以大型托管实例组为目标。使用 AWS CLI、AWS Tools for Windows PowerShell 或 Systems Manager 开发工具包，您可以通过指定多个标签来标识目标。

4. 然后将这些信息 (计划、目标、文档、参数) 绑定到托管实例。

此信息与目标的绑定称为创建关联。您可以使用 Amazon EC2 控制台、AWS CLI、AWS Tools for Windows PowerShell 或 AWS 开发工具包来创建关联。

5. 在发送创建关联的请求后，关联的状态将显示“待处理”。系统将尝试访问所有目标并立即应用关联中指定的状态。

Note

如果您创建计划在之前的关联仍在运行时运行的新关联，那么之前的关联将超时，系统会运行新关联。

6. Systems Manager 为请求的每个目标实例报告请求的状态。

您可以在 EC2 控制台中或使用 [DescribeInstanceAssociationsStatus](#) API 操作查看状态详细信息。如果在创建关联时选择将命令输出写入到 S3，则还可以在指定的 Amazon S3 存储桶中查看此输出。

7. 创建关联之后，状态管理器 将根据关联中定义的计划重新应用状态。

您可以更新您的关联文档并根据需要重新应用它们。您也可以创建多个关联版本。

示例 状态管理器 文档

此部分包含示例 状态管理器 文档。出于演示目的，我们对示例进行了简化。有关创建自定义文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

示例 1：使用“aws:runPowerShellScript”插件获取实例名称

以下文档包含一个调用 `aws:runPowerShellScript` 插件以返回实例主机名的步骤。此文档可在 Windows 和 Linux 实例上运行。有关 Systems Manager 插件的更多信息，请参阅[顶级元素 \(p. 312\)](#)。

```
{
  "schemaVersion": "2.2",
  "description": "Sample document",
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "runPowerShellScript",
      "inputs": {
        "runCommand": [
          "hostname"
        ]
      }
    }
  ]
}
```

示例 2：使用“aws:runShellScript”插件获取实例名称

以下文档包含一个调用 `aws:runShellScript` 插件以返回实例主机名的步骤。此文档只能在 Linux 实例上运行。

```
{
  "schemaVersion": "2.2",
  "description": "Sample document",
  "mainSteps": [
    {
      "action": "aws:runShellScript",
      "name": "runShellScript",
      "inputs": {
        "runCommand": [
          "hostname"
        ]
      }
    }
  ]
}
```

示例 3：使用“aws:cloudWatch”插件启用 CloudWatch Logs

以下文档包含一个调用 `aws:cloudWatch` 插件启用 Amazon CloudWatch Logs 的步骤。此文档只能在 Windows 实例上运行。

```
{
  "schemaVersion": "2.2",
  "description": "Sample document",
  "mainSteps": [
```



```
{
  "action": "aws:cloudWatch",
  "name": "cloudWatch",
  "settings": {
    "startType": "Enabled"
  }
}
```

示例 4：使用“aws:runPowerShellScript”和“aws:runShellScript”插件获取实例名称

以下文档包含两个调用 `aws:runPowerShellScript` 和 `aws:runShellScript` 插件以返回实例主机名的步骤。此文档只能在 Linux 实例上运行。

```
{
  "schemaVersion": "2.2",
  "description": "Sample document",
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "runPowerShellScript",
      "inputs": {
        "runCommand": [
          "hostname"
        ]
      }
    },
    {
      "action": "aws:runShellScript",
      "name": "runShellScript",
      "inputs": {
        "runCommand": [
          "hostname"
        ]
      }
    }
  ]
}
```

在 Systems Manager 中使用关联

在状态管理器中，关联是绑定配置信息的结果，它定义您希望实例进入实例本身的状态。此信息指定何时以及如何运行与实例相关的操作，以确保您的 Amazon EC2 和混合基础设施处于预期或一致的状态。

请参阅以下主题，了解如何在状态管理器中创建和管理关联。

主题

- [创建关联 \(控制台\) \(p. 280\)](#)
- [编辑并创建新的关联版本 \(控制台\) \(p. 282\)](#)
- [使用 Targets 参数创建关联 \(CLI\) \(p. 283\)](#)
- [查看关联历史记录 \(p. 283\)](#)

创建关联 (控制台)

此部分介绍如何使用 Amazon EC2 控制台创建状态管理器关联。此部分中的示例说明如何基于自定义 SSM 文档创建关联。如果这是您首次创建关联，建议在测试环境中执行此过程。有关使用 AWS CLI 创建关联的示例，请参阅[演练：自动更新 SSM 代理 \(CLI\) \(p. 286\)](#)。

开始前的准备工作

在完成以下过程之前，请确认您至少有一个为 Systems Manager 配置的实例正在运行。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

创建 状态管理器 关联 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择创建文档。
4. 在 Name 字段中，键入一个将此文档标识为 状态管理器 的测试文档的描述性名称。
5. 在文档类型列表中，选择命令文档。
6. 在内容区域中：
 - 选择 JSON 旁边的按钮。
 - 删除 Content 字段中预先填充的括号 {}，然后将以下示例文档复制并粘贴到 Content 字段中。

此文档包含一个调用 aws:runPowerShellScript 插件以返回实例主机名的步骤。此文档可在 Windows 实例上运行。

```
{
  "schemaVersion": "2.0",
  "description": "Sample document",
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "runPowerShellScript",
      "inputs": {
        "runCommand": [
          "hostname"
        ]
      }
    ]
  ]
}
```

7. 在导航窗格中，选择 状态管理器。
8. 选择创建关联。
9. 在名称框中，为此关联键入一个描述性名称。例如，将关联命名为 **TestHostnameAssociation**。
10. 在命令文档列表中，选择刚才创建的文档。
11. 在文档版本列表中，保留默认值。
12. 忽略 Parameters 部分，因为测试文档不采用参数。
13. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

14. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
15. 忽略 Output options 部分。下面的过程介绍在 S3 存储桶中启用命令输出存储的步骤：[编辑并创建新的关联版本 \(控制台\)](#) (p. 282)。
 16. 选择创建关联。系统将尝试在实例上创建关联并立即应用状态。在此情况下，创建关联后，系统将尝试返回主机名。关联状态显示 Pending。
 17. 选择浏览器的刷新按钮。状态将变为 Success。

您无法查看此过程的输出 (实例名称)，因为它写入到了 Amazon S3。

有关编辑关联、将输出写入 Amazon S3 存储桶和查看实例名称的信息，请参阅下一个过程[编辑并创建新的关联版本 \(控制台\)](#) (p. 282)。

编辑并创建新的关联版本 (控制台)

您可以编辑关联以指定新名称、计划或目标。您也可以选择将命令输出写入到 Amazon S3 存储桶。编辑关联后，Systems Manager 将创建新版本。您可在编辑后查看不同的版本，如以下过程中所述。

Note

此过程要求您具有对现有 S3 存储桶的写入权限。如果您之前未使用 S3，请注意使用 S3 会产生费用。有关如何创建存储桶的信息，请参阅[创建存储桶](#)。

编辑 状态管理器 关联 ()

1. 在导航窗格中，选择 状态管理器。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 状态管理器。

2. 选择在上一个过程中创建的关联，然后选择编辑。
3. 在 Name 字段中，键入一个新名称。例如，键入 **TestHostnameAssociation2**。
4. 在指定计划部分中，选择一个新选项。例如，选择 Cron 计划生成器，然后选择每 1 小时。
5. (可选) 要将命令输出写入到 Amazon S3 存储桶，请在输出选项部分中执行以下操作：
 - 选择 Enable writing output to S3。
 - 在 S3 存储桶名称字段中，键入您有写入权限的 S3 存储桶的名称。
 - (可选) 要将输出写入存储桶中的文件夹，请在 S3 键前缀字段中输入其名称。如果您指定名称的文件夹不存在，状态管理器 将创建该文件夹。
6. 选择编辑关联。
7. 在关联页面中，选择刚才编辑的关联的名称，然后选择版本选项卡。系统将列出您已创建和编辑的关联的每个版本。
8. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
9. 选择您指定用于存储命令输出的 S3 存储桶的名称，然后选择以运行关联的实例的 ID 命名的文件夹。(如果您选择将输出存储在存储桶中的文件夹内，请先打开它。)
10. 下拉多个级别至 stdout 文件夹中的 awsruntimePowerShell 文件。

11. 选择打开或下载查看主机名。

使用 Targets 参数创建关联 (CLI)

利用 `targets` 参数，您可以在数十、数百或数千个实例上创建关联。`targets` 参数支持基于您为实例指定的 Amazon EC2 标签 `Key, Value` 组合。在您运行创建关联的请求时，系统会在符合指定条件的所有实例上查找并尝试创建关联。有关 `targets` 参数的更多信息，请参阅 [将命令发送到队列](#) (p. 185)。有关 Amazon EC2 标签的更多信息，请参阅 Amazon EC2 用户指南 中的 [标记 Amazon EC2 资源](#)。

以下 AWS CLI 示例显示了如何在创建关联时使用 `targets` 参数。

```
aws ssm create-association --targets Key=tag:TagKey,Values=TagValue --name AWS-UpdateSSMAgent --schedule "cron(0 0 2 ? * SUN *)"
```

为名为 "ws-0123456789012345" 的托管实例创建关联

```
aws ssm create-association --association-name value --targets Key=InstanceId,Values=ws-0123456789 --name AWS-UpdateSSMAgent --schedule "cron(0 0 2 ? * SUN *)"
```

Note

如果您从与文档关联的标记实例组中删除一个实例，则该实例将不再与文档相关联。

查看关联历史记录

您可以使用 [DescribeAssociationExecutions](#) API 操作查看特定关联 ID 的所有执行。此操作可让您快速查看状态管理器 关联的状态、详细状态、结果、上次执行时间以及其他信息。此 API 操作还包括筛选器，帮助您根据指定的条件快速找到关联。例如，您可以指定一个确切的日期和时间，并使用 `GREATER_THAN` 筛选器以仅查看在该指定的日期和时间之后处理的那些执行。

例如，如果一个关联执行失败，则可使用 [DescribeAssociationExecutionTargets](#) API 操作来深入了解特定执行的详细信息。此操作将为您显示资源，例如实例 ID、关联运行的位置和各种关联状态。随后，您可以快速查看哪些资源或实例无法运行关联。利用资源 ID，您随后可以查看命令执行详细信息，以准确了解命令中的哪个步骤已失败。

此部分中的示例还包括有关如何使用 [StartAssociationsOnce](#) API 操作立即运行且仅运行一次关联的信息。在调查失败的关联执行时，可以使用此 API 操作。如果发现一个关联失败，则可以在资源上进行更改，然后立即运行关联以查看对资源所做的更改是否能让关联成功运行。

使用 Systems Manager 控制台查看关联历史记录

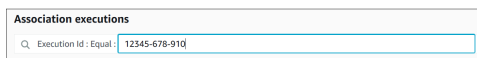
使用以下过程可查看特定关联 ID 的执行历史记录，然后查看一个或多个资源的执行详细信息。

查看特定关联 ID 的执行历史记录

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 选择 状态管理器。
3. 在 Association id (关联 ID) 字段中，选择要查看其历史记录的关联。
4. 选择 View details (查看详细信息) 按钮。
5. 选择 Execution history (执行历史记录) 选项卡。
6. 选择要查看其资源级别执行详细信息的关联。例如，选择显示 Failed (失败) 状态的关联。然后，您可以查看无法运行关联的实例的执行详细信息。

Note

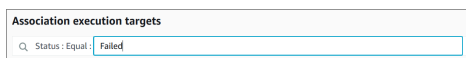
使用搜索框筛选条件以找到要查看其详细信息的执行。



7. 选择执行 ID。此时将打开 Association execution targets (关联执行目标) 页面。此页面显示所有已运行关联的资源。
8. 选择一个资源 ID 以查看有关此资源的特定信息。

Note

使用搜索框筛选条件以找到要查看其详细信息的资源。



9. 如果您正在调查无法运行的关联，则可使用 Apply association now (立即应用关联) 按钮来立即运行且仅运行一次关联。在对无法运行关联的资源进行更改后，在导航位置提示中选择 Association ID (关联 ID) 链接。
10. 选择 Apply association now (立即应用关联) 按钮。在执行完成后，验证关联执行是否成功。

使用 AWS CLI 查看关联历史记录

使用以下过程可查看特定关联 ID 的执行历史记录，然后查看一个或多个资源的执行详细信息。

查看特定关联 ID 的执行历史记录

1. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令以查看特定关联 ID 的执行列表。此命令包含一个筛选器，可将结果限定为仅在特定日期和时间之后发生的那些执行。如果您希望查看特定关联 ID 的所有执行，请删除 `--filter parameter`。

```
aws ssm describe-association-executions --association-id ID --filters
Key=CreateTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN
```

系统将返回类似于以下内容的信息。

```
{
  "AssociationExecutions": [
    {
      "Status": "Success",
      "DetailedStatus": "Success",
      "AssociationId": "12345-abcdef-6789-ghij",
      "ExecutionId": "abcde-12345-fghi-6789",
      "CreateTime": 1523986028.219,
      "AssociationVersion": "1"
    },
    {
      "Status": "Success",
      "DetailedStatus": "Success",

```

```
{
  "AssociationId": "12345-abcdef-6789-ghij",
  "ExecutionId": "zzzz-333-xxxx-4444",
  "CreatedTime": 1523984226.074,
  "AssociationVersion": "1"
},
{
  "Status": "Success",
  "DetailedStatus": "Success",
  "AssociationId": "12345-abcdef-6789-ghij",
  "ExecutionId": "4545-a4a4a4-3636",
  "CreatedTime": 1523982404.013,
  "AssociationVersion": "1"
}
]
```

您可以使用一个或多个筛选条件来限制结果。以下示例返回在特定日期和时间之前运行的所有关联。

```
aws ssm describe-association-executions --association-id ID --filters
  Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=LESS_THAN
```

下面返回在特定日期和时间之后成功运行的所有关联。

```
aws ssm describe-association-executions --association-id ID --filters
  Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN
  Key=Status,Value=Success,Type=EQUAL
```

4. 执行以下命令可查看已运行特定执行的所有目标。

```
aws ssm describe-association-execution-targets --association-id ID --execution-id ID
```

您可以使用一个或多个筛选条件来限制结果。以下示例返回有关所有无法运行特定关联的目标的信息。

```
aws ssm describe-association-execution-targets --association-id ID --execution-id ID --
filters Key=Status,Value="Failed"
```

以下示例返回有关无法运行关联的特定托管实例的信息。

```
aws ssm describe-association-execution-targets --association-id ID --execution-
id ID --filters Key=Status,Value=Failed Key=ResourceId,Value="instance ID"
  Key=ResourceType,Value=ManagedInstance
```

5. 如果您正在调查无法运行的关联，则可使用 [StartAssociationsOnce](#) API 操作来立即运行且仅运行一次关联。在对无法运行关联的资源进行更改后，运行以下命令可立即运行且仅运行一次关联。

```
aws ssm start-associations-once --association-id ID
```

Systems Manager 状态管理器 演练

以下演练演示如何使用 Amazon EC2 控制台或 AWS CLI 创建和配置 状态管理器 关联。这些演练还演示如何使用 状态管理器 自动执行常见管理任务。

主题

- [演练：自动更新 SSM 代理 \(CLI\) \(p. 286\)](#)
- [演练：在 EC2 Windows 实例上自动更新半虚拟化驱动程序 \(控制台\) \(p. 287\)](#)

演练：自动更新 SSM 代理 (CLI)

以下步骤将指导您完成使用 AWS Command Line Interface (AWS CLI) 创建 状态管理器 关联的过程。关联会根据您指定的计划自动更新 SSM 代理。有关 SSM 代理的更多信息，请参阅[安装和配置 SSM 代理 \(p. 14\)](#)。

要查看有关不同版本 SSM 代理的详细信息，请参阅[发行说明](#)。

开始前的准备工作

在完成以下过程之前，请确认您至少有一个为 Systems Manager 配置的 Amazon EC2 实例 (Linux 或 Windows) 正在运行。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

创建关联以实现 SSM 代理 的自动更新

1. 在您的本地计算机上，[下载](#)最新版本的 AWS CLI。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令，通过使用 Amazon EC2 标签将实例设为目标来创建关联。Schedule 参数可以设置一个计划，以在每周日凌晨 2:00 (UTC) 运行关联。

```
aws ssm create-association --targets Key=tag:TagKey,Values=TagValue --name AWS-UpdateSSMAgent --schedule-expression "cron(0 0 2 ? * SUN *)"
```

Note

状态管理器 关联不支持所有的 cron 和 rate 表达式。有关为关联创建 cron 和 rate 表达式的更多信息，请参阅[参考：Systems Manager 的 Cron 和 Rate 表达式 \(p. 418\)](#)。

如果需要，您也可以通过在以逗号分隔的列表中指定实例 ID 来将多个实例设为目标。

```
aws ssm create-association --targets
  Key=instanceids,Values=InstanceID,InstanceID,InstanceID --name your document name --
schedule-expression "cron(0 0 2 ? * SUN *)"
```

系统将返回类似于以下内容的信息。

```
{
  "AssociationDescription": {
    "ScheduleExpression": "cron(0 0 2 ? * SUN *)",
    "Name": "AWS-UpdateSSMAgent",
    "Overview": {
      "Status": "Pending",
      "DetailedStatus": "Creating"
    },
    "AssociationId": "123.....",
    "DocumentVersion": "$DEFAULT",
    "LastUpdateAssociationDate": 1504034257.98,
  }
}
```



```
{
  "Date": 1504034257.98,
  "AssociationVersion": "1",
  "Targets": [
    {
      "Values": [
        "TagValue"
      ],
      "Key": "tag:TagKey"
    }
  ]
}
```

系统将尝试在实例上创建关联并立即应用状态。关联状态显示 Pending。

4. 运行以下命令查看您刚才创建的关联的更新状态。

```
aws ssm list-associations
```

Note

如果您的实例目前正在运行 SSM 代理的最新版本，则状态将显示 Failed。这是预期行为。发布 SSM 代理的新版本时，关联将自动安装新代理，而且状态将显示 Success。

演练：在 EC2 Windows 实例上自动更新半虚拟化驱动程序 (控制台)

Amazon Windows AMI 包含一系列驱动程序，以允许访问虚拟化硬件。Amazon EC2 会使用这些驱动程序将实例存储和 Amazon EBS 卷映射到其设备。我们建议您安装最新的驱动程序来提高您的 EC2 Windows 实例的稳定性和性能。有关半虚拟化驱动程序的更多信息，请参阅 [AWS 半虚拟化驱动程序](#)。

以下演练介绍如何配置 状态管理器 关联以在新的 AWS 半虚拟化驱动程序可用时自动下载和安装这些驱动程序。

开始前的准备工作

在完成以下过程之前，请确认您至少有一个为 Systems Manager 配置的 Amazon EC2 Windows 实例正在运行。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

Note

以下过程描述了您在 Amazon EC2 控制台中执行的步骤。您也可以在新的 [AWS Systems Manager 控制台](#) 中执行这些步骤。新控制台中的步骤将不同于下面的步骤。

创建自动更新半虚拟化驱动程序的 状态管理器 关联

1. 打开 [Amazon EC2 控制台](#)，在导航窗格中展开 Systems Manager Services，然后选择 状态管理器。
2. 选择创建关联。
3. 在 Association Name 字段中，键入一个描述性名称。
4. 在 Select Document 列表中，选择 AWS-ConfigureAWSPackage。
5. 在 Select Targets by 部分中，选择选项。

Note

如果您选择使用标签将实例设为目标，并指定映射到 Linux 实例的标签，则关联在 Windows 实例上将成功，但在 Linux 实例上将失败。关联的总体状态将显示 Failed。

6. 在计划部分，选择一个选项。更新的半虚拟化驱动程序每年仅发布几次，因此如果需要，您可以计划每月运行一次关联。

7. 在 Parameters 部分中，选择 Action 列表中的 Install。
8. 从 Name 列表中，选择 AWSPVDriver。您可以将 Version 字段留空。
9. 如果您要向 Amazon S3 存储桶中写入关联详细信息，请在 Advanced 部分中，选择 Write to S3。
10. 忽略 S3Region 字段。此字段已弃用。在 S3Bucket Name 字段中指定存储桶的名称。如果要输出写入子文件夹，请在 S3Key Prefix 字段中指定子文件夹名称。
11. 选择 Create Association，然后选择 Close。系统将尝试在实例上创建关联并立即应用状态。关联状态显示 Pending。
12. 在 Association 页面的右角，选择刷新按钮。如果您在一个或多个 EC2 Windows 实例上创建关联，状态将更改为 Success。如果未为 Systems Manager 正确配置实例，或您无意中将 Linux 实例设为了目标，则状态将显示 Failed。
13. 如果状态为 Failed，则选择 Instances 选项卡并验证关联是否已在 EC2 Windows 实例上成功创建。如果 Windows 实例显示失败状态，请验证 SSM 代理是否在实例上运行，并验证实例是否配置有适用于 Systems Manager 的 IAM 角色。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

AWS Systems Manager 共享资源

Systems Manager 使用以下共享资源来管理和配置您的 AWS 资源。

主题

- [AWS Systems Manager 托管实例 \(p. 289\)](#)
- [AWS Systems Manager 激活 \(p. 289\)](#)
- [AWS Systems Manager 文档 \(p. 289\)](#)
- [AWS Systems Manager Parameter Store \(p. 361\)](#)

AWS Systems Manager 托管实例

托管实例 是为 AWS Systems Manager 配置的任何计算机。您可以将 Amazon EC2 实例或混合环境中的本地计算机配置为托管实例。Systems Manager 支持各种 Linux 发行版，包括 Raspberry Pi 设备和 Microsoft Windows。

在控制台中，带有“mi-”前缀的所有计算机都是本地服务器或虚拟机 (VM) 托管实例。

有关 Systems Manager 先决条件的信息，请参阅[Systems Manager 先决条件 \(p. 6\)](#)。有关将本地服务器和 VM 配置为托管实例的信息，请参阅[在混合环境中设置 AWS Systems Manager \(p. 31\)](#)。

Note

Systems Manager 需要准确的时间引用以便执行其操作。如果实例的日期和时间设置不正确，它们可能与 API 请求的签名日期不匹配。有关更多信息，请参阅[使用案例和最佳实践 \(p. 424\)](#)。

AWS Systems Manager 激活

要在混合环境中将服务器和虚拟机 (VM) 设置为托管实例，请创建托管实例激活。完成激活后，您将收到一个激活代码和 ID。此代码/ID 组合具有 Amazon EC2 访问 ID 和私有密钥的功能，可提供从托管实例对 Systems Manager 服务的安全访问。有关将本地服务器和 VM 配置为托管实例的信息，请参阅[在混合环境中设置 AWS Systems Manager \(p. 31\)](#)。

AWS Systems Manager 文档

AWS Systems Manager 文档 (SSM 文档) 定义 Systems Manager 对您的托管实例执行的操作。Systems Manager 包括十多个预先配置的文档，可以通过在运行时指定参数进行使用。文档使用 JavaScript Object Notation (JSON) 或 YAML，并包括您指定的步骤和参数。

SSM 文档类型

下表介绍了不同类型的 SSM 文档。

类型	一起使用	详细信息
命令文档	Run Command (p. 171) 状态管理器 (p. 277)	Run Command 使用命令文档执行命令。状态管理器 使用命令文档应用配置。在实例生命周期内的任何时刻，这些操作都可在一个或多个目标上运行，

类型	一起使用	详细信息
策略文档	状态管理器 (p. 277)	策略文档对您的目标执行策略。如果删除策略文档，将不再执行策略操作 (例如，收集清单)。
自动化文档	自动化 (p. 100)	在执行常规维护和部署任务时 – 如创建或更新 Amazon 系统映像 (AMI) – 使用自动化文档。

SSM 文档版本与执行

您可以创建并保存不同版本的文档。您可以为每个文档指定默认版本。文档的默认版本可以更新到较新版本或恢复到较旧版本。当您更改文档的内容时，Systems Manager 将自动递增文档的版本。您可以检索并使用先前版本的文档。

自定义文档

如果您要通过文档自定义步骤和操作，可以自行创建文档。当您首次使用某一文档对实例执行操作时，系统会将此文档存储在您的 AWS 账户中。有关如何创建 Systems Manager 文档的更多信息，请参阅[创建 Systems Manager 文档 \(p. 297\)](#)。

为文档添加标签

您可以为文档添加标签，以便根据为其分配的标签快速识别一个或多个文档。例如，您可以为特定环境、部门、用户、组或时间段的文档添加标签。此外，您还可以通过创建一个指定用户或组可访问的标签的 IAM 策略来限制对文档的访问。有关更多信息，请参阅[为 Systems Manager 文档添加标签 \(p. 299\)](#)。

共享文档

您可以将文档公开，或者与特定 AWS 账户共享。有关更多信息，请参阅[共享 Systems Manager 文档 \(p. 302\)](#)。

SSM 文档限制

有关 SSM 文档限制的信息，请参阅[AWS Systems Manager 限制](#)。

主题

- [Systems Manager 预定义文档 \(p. 290\)](#)
- [SSM 文档架构和功能 \(p. 291\)](#)
- [SSM 文档语法 \(p. 292\)](#)
- [创建 Systems Manager 文档 \(p. 297\)](#)
- [为 Systems Manager 文档添加标签 \(p. 299\)](#)
- [共享 Systems Manager 文档 \(p. 302\)](#)
- [创建复合文档 \(p. 308\)](#)
- [从远程位置运行文档 \(p. 309\)](#)
- [SSM 文档插件参考 \(p. 312\)](#)
- [Systems Manager 自动化文档参考 \(p. 334\)](#)

Systems Manager 预定义文档

为了帮助您快速入门，Systems Manager 提供了预定义文档。要在 AWS Systems Manager 控制台中查看这些文档，请在左侧导航窗格中选择文档。选择某个文档后，选择查看详细信息以查看所选文档的相关信息。

您还可以使用 AWS CLI 和 Tools for Windows PowerShell 命令查看一组文档，并获得这些文档的描述。

要使用 AWS CLI 查看有关文档的信息，请运行以下命令：

```
aws ssm list-documents
```

```
aws ssm describe-document --name "document_name"
```

要使用 Tools for Windows PowerShell 查看有关文档的信息，请运行以下命令：

```
Get-SSMDocumentList
```

```
Get-SSMDocumentDescription -Name "document_name"
```

SSM 文档架构和功能

Systems Manager 文档当前使用以下架构版本。

- Command 型文档可以使用 1.2、2.0 和 2.2 版架构。如果您目前使用 1.2 版架构文档，我们建议您创建使用 2.2 版架构的文档。
- Policy 型文档必须使用 2.0 版或更高版本的架构。
- Automation 型文档必须使用 0.3 版架构。
- 您可以创建 JSON 或 YAML 格式的文档。

通过为 Command 和 Policy 文档使用最新架构版本，您可以利用以下功能。

2.2 版架构文档功能

功能	详细信息
编辑文档	文档现在可以更新。如果版本为 1.2，对文档的任何更新都需要另存为其他名称。
自动版本控制	对文档的任何更新都会创建一个新版本。这不是架构版本，而是文档版本。
默认版本	如果您拥有某个文档的多个版本，可以指定哪个版本是默认文档。
顺序	文档中的插件或步骤按指定的顺序执行。
跨平台支持	跨平台支持可让您为同一个 SSM 文档中的不同插件指定不同操作系统。跨平台支持在某个步骤中使用 <code>precondition</code> 参数。

Note

您必须将实例上的 SSM 代理更新到最新版本才能使用新的 Systems Manager 功能和 SSM 文档功能。有关更多信息，请参阅 [示例：更新 SSM 代理 \(p. 183\)](#)。

下表列出了主要架构版本之间的区别。

版本 1.2	版本 2.2 (最新版本)	详细信息
runtimeConfig	mainSteps	在版本 2.2 中，mainSteps 部分替换了 runtimeConfig 部分。mainSteps 部分启用了 Systems Manager，以便依顺序执行步骤。
属性	inputs	在版本 2.2 中，inputs 部分替换了 properties 部分。inputs 部分接受步骤参数。
命令	runCommand	在版本 2.2 中，inputs 部分接收 runCommand 参数，而不是 commands 参数。
id	action	在版本 2.2 中，Action 替换了 ID。这只是更改了名称。
不适用	name	在版本 2.2 中，name 是用户定义的任意步骤名称。

使用前提条件参数

在 2.2 版或更高版本架构中，您可以使用 `precondition` 参数为每个插件指定目标操作系统。`precondition` 参数支持 `platformType` 和值 `Windows` 或 `Linux`。

对于使用 2.2 版或更高版本架构的文档，如果未指定 `precondition`，每个插件将被执行或跳过，具体取决于插件与操作系统的兼容性。对于使用 2.0 版或更低版本架构的文档，不兼容的插件将会引发错误。

例如，在 2.2 版架构文档中，如果未指定 `precondition`，将会列出 `aws:runShellScript` 插件，该步骤在 Linux 实例上运行，但系统会在 Windows 实例上跳过该步骤，因为 `aws:runShellScript` 与 Windows 实例不兼容。但是，对于 2.0 版架构文档，如果您指定 `aws:runShellScript` 插件，然后在 Windows 实例上运行文档，执行将会失败。本部分稍后将介绍在 SSM 文档中使用前提条件参数的示例。

SSM 文档语法

文档语法由用于创建文档的架构版本定义。我们建议您使用 2.2 版或更高版本的架构。使用此架构版本的文档包含以下顶级元素。有关可在这些元素中指定的属性的信息，请参阅[顶级元素 \(p. 312\)](#)。

- `schemaVersion`：要使用的架构版本。
- `Description`：您提供的描述文档目的的信息。
- `Parameters`：文档接受的参数。对于您经常引用的参数，我们建议您在 Systems Manager Parameter Store 中存储这些参数，然后进行引用。您可以在文档的这一部分引用 `String` 和 `StringList` Systems Manager 参数。您不能在文档的这一部分引用 `Secure String` Systems Manager 参数。有关更多信息，请参阅[AWS Systems Manager Parameter Store \(p. 361\)](#)。
- `mainSteps`：可以包含多个步骤 (插件) 的对象。步骤包括一个或多个操作、可选前提条件、各操作的唯一名称以及这些操作的 `inputs` (参数)。有关支持的插件和插件属性的列表，请参阅[SSM 文档插件参考 \(p. 312\)](#)。

Important

操作的名称不能包含空格。如果名称包含空格，您将收到 `InvalidDocumentContent` 错误。

主题

- 架构版本 2.2 (p. 293)
- 架构版本 1.2 (p. 296)

架构版本 2.2

以下示例以 JSON 格式显示 2.2 版架构文档的顶级元素。

```
{
  "schemaVersion": "2.2",
  "description": "A description of the document.",
  "parameters": {
    "parameter 1": {
      "one or more parameter properties"
    },
    "parameter 2": {
      "one or more parameter properties"
    },
    "parameter 3": {
      "one or more parameter properties"
    }
  },
  "mainSteps": [
    {
      "action": "plugin 1",
      "name": "A name for this action.",
      "inputs": {
        "name": "{{ input 1 }}",
        "name": "{{ input 2 }}",
        "name": "{{ input 3 }}"
      }
    }
  ]
}
```

YAML 架构版本 2.2 示例

您可以将以下 YAML 文档用于 Run Command 以返回一个或多个实例的主机名称。

```
---
schemaVersion: '2.2'
description: Sample document
mainSteps:
- action: aws:runPowerShellScript
  name: runPowerShellScript
  inputs:
    runCommand:
    - hostname
```

架构版本 2.2 前提条件参数示例

2.2 版架构提供跨平台支持。这意味着在一个 SSM 文档内，您可以为不同的插件指定不同的操作系统。跨平台支持在某个步骤中使用 `precondition` 参数，如下例所示。

```
{
  "schemaVersion": "2.2",
  "description": "cross-platform sample",
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
```

```
      "name": "PatchWindows",
      "precondition": {
        "StringEquals": [
          "platformType",
          "Windows"
        ]
      },
      "inputs": {
        "runCommand": [
          "cmds"
        ]
      }
    },
    {
      "action": "aws:runShellScript",
      "name": "PatchLinux",
      "precondition": {
        "StringEquals": [
          "platformType",
          "Linux"
        ]
      },
      "inputs": {
        "runCommand": [
          "cmds"
        ]
      }
    }
  ]
}
```

架构版本示例 2.2

您可以将以下 YAML 文档用于 状态管理器 以下载并安装 ClamAV 防病毒软件。状态管理器 强制实施特定配置，这意味着每次运行 状态管理器 关联时，系统会检查是否安装了 ClamAV 软件。如果未安装，状态管理器 会重新运行本文档。

```
---
schemaVersion: '2.2'
description: State Manager Bootstrap Example
parameters: {}
mainSteps:
- action: aws:runShellScript
  name: configureServer
  inputs:
    runCommand:
      - sudo yum install -y httpd24
      - sudo yum --enablerepo=epel install -y clamav
```

架构版本 2.0 YAML 清单示例

您可以将以下 YAML 文档用于 状态管理器 以收集关于实例的清单元数据。

```
---
schemaVersion: '2.2'
description: Software Inventory Policy Document.
parameters:
  applications:
    type: String
    default: Enabled
    description: "(Optional) Collect data for installed applications."
    allowedValues:
```

```

- Enabled
- Disabled
awsComponents:
  type: String
  default: Enabled
  description: "(Optional) Collect data for AWS Components like amazon-ssm-agent."
  allowedValues:
    - Enabled
    - Disabled
networkConfig:
  type: String
  default: Enabled
  description: "(Optional) Collect data for Network configurations."
  allowedValues:
    - Enabled
    - Disabled
windowsUpdates:
  type: String
  default: Enabled
  description: "(Optional) Collect data for all Windows Updates."
  allowedValues:
    - Enabled
    - Disabled
instanceDetailedInformation:
  type: String
  default: Enabled
  description: "(Optional) Collect additional information about the instance, including
    the CPU model, speed, and the number of cores, to name a few."
  allowedValues:
    - Enabled
    - Disabled
customInventory:
  type: String
  default: Enabled
  description: "(Optional) Collect data for custom inventory."
  allowedValues:
    - Enabled
    - Disabled
mainSteps:
- action: aws:softwareInventory
  name: collectSoftwareInventoryItems
  inputs:
    applications: "{{ applications }}"
    awsComponents: "{{ awsComponents }}"
    networkConfig: "{{ networkConfig }}"
    windowsUpdates: "{{ windowsUpdates }}"
    instanceDetailedInformation: "{{ instanceDetailedInformation }}"
    customInventory: "{{ customInventory }}"

```

架构版本 2.2 AWS-ConfigureAWSPackage 示例

以下示例显示 AWS-ConfigureAWSPackage 文档。mainSteps 部分在 action 步骤中包括 aws:configurePackage 插件。

```

{
  "schemaVersion": "2.2",
  "description": "Install or uninstall the latest version or specified version of an AWS
  package.
    Available packages include the following: AWSPVDriver,
  AwsEnaNetworkDriver, IntelSriovDriver,
    AwsVssComponents, and AmazonCloudWatchAgent, and AWSSupport-EC2Rescue.",
  "parameters": {
    "action": {
      "description": "(Required) Specify whether or not to install or uninstall the package.",

```



```

    "type": "String",
    "allowedValues": [
      "Install",
      "Uninstall"
    ]
  },
  "name": {
    "description": "(Required) The package to install/uninstall.",
    "type": "String",
    "allowedPattern": "^arn:[a-z0-9][-.a-z0-9]{0,62}:[a-z0-9][-.a-z0-9]{0,62}:([a-z0-9]
    [-.a-z0-9]{0,62})?:([a-z0-9][-.a-z0-9]{0,62})?:package\\/[a-zA-Z][a-
    zA-Z0-9\\-_{0,39}$|^
    [a-zA-Z][a-zA-Z0-9\\-_{0,39}$"
  },
  "version": {
    "description": "(Optional) A specific version of the package to install or uninstall. If
    installing,
    the system installs the latest published version, by default. If uninstalling, the
    system uninstalls
    the currently installed version, by default. If no installed version is found, the
    latest published
    version is downloaded, and the uninstall action is run.",
    "type": "String",
    "default": "latest"
  }
},
"mainSteps": [{
  "action": "aws:configurePackage",
  "name": "configurePackage",
  "inputs": {
    "name": "{{ name }}",
    "action": "{{ action }}",
    "version": "{{ version }}"
  }
}]
}

```

架构版本 1.2

以下示例显示 1.2 版架构文档的顶级元素。

```

{
  "schemaVersion": "1.2",
  "description": "A description of the Systems Manager document.",
  "parameters": {
    "parameter 1": {
      "one or more parameter properties"
    },
    "parameter 2": {
      "one or more parameter properties"
    },
    "parameter 3": {
      "one or more parameter properties"
    }
  },
  "runtimeConfig": {
    "plugin 1": {
      "properties": [
        {
          "one or more plugin properties"
        }
      ]
    }
  }
}

```

```
}
```

1.2 版架构示例

以下示例显示 AWS-RunShellScript Systems Manager 文档。runtimeConfig 部分包含 aws:runShellScript 插件。

```
{
  "schemaVersion": "1.2",
  "description": "Run a shell script or specify the commands to run.",
  "parameters": {
    "commands": {
      "type": "StringList",
      "description": "(Required) Specify a shell script or a command to run.",
      "minItems": 1,
      "displayType": "textarea"
    },
    "workingDirectory": {
      "type": "String",
      "default": "",
      "description": "(Optional) The path to the working directory on your instance.",
      "maxChars": 4096
    },
    "executionTimeout": {
      "type": "String",
      "default": "3600",
      "description": "(Optional) The time in seconds for a command to complete before it is considered to have failed. Default is 3600 (1 hour). Maximum is 172800 (48 hours).",
      "allowedPattern": "([1-9][0-9]{0,3})|(1[0-9]{1,4})|(2[0-7][0-9]{1,3})|(28[0-7][0-9]{1,2})|(28800)"
    }
  },
  "runtimeConfig": {
    "aws:runShellScript": {
      "properties": [
        {
          "id": "0.aws:runShellScript",
          "runCommand": "{{ commands }}",
          "workingDirectory": "{{ workingDirectory }}",
          "timeoutSeconds": "{{ executionTimeout }}"
        }
      ]
    }
  }
}
```

创建 Systems Manager 文档

如果 Systems Manager 公有文档限制了您希望对托管实例执行的操作，您可以自行创建文档。创建新文档时，我们建议使用架构版本 2.2 或更高版本。

开始前的准备工作

在创建 SSM 文档之前，我们建议您阅读有关 SSM 文档可用的其他架构、功能和语法的信息。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

Note

如果计划创建适用于 状态管理器 的 SSM 文档；请注意以下详细信息：

- 通过创建使用不同文档的不同 状态管理器 关联，您可以将多个文档分配给一个目标。

- 只要有权限，您就可以将共享文档用于状态管理器，但是您无法将共享文档与实例关联。如果您想要使用或分享与一个或多个目标相关联的文档，则您必须先创建文档副本，然后才能使用或分享。
- 如果您创建的文档含有互相冲突的插件（例如加入域的插件和从域中删除的插件），则最终状态取决于最后执行的插件。状态管理器不会验证您的文档中的命令或插件的逻辑顺序或合理性。
- 处理文档时，系统首先应用实例关联，然后应用标记实例组关联。如果实例处于多个标记实例组中，则对应标记实例组的文档不会以任何特定顺序执行。如果实例通过其实例 ID 直接对应多个文档，那么也不存在特定的执行顺序。
- 如果您更改状态管理器文档的默认版本，则下次 Systems Manager 将关联应用到实例时，使用该文档的所有关联都将开始使用新的默认版本。

如果您创建适用于状态管理器的 SSM 文档，在将其添加到系统后，必须将该文档与您的托管实例相关联。有关更多信息，请参阅 [创建关联 \(控制台\)](#) (p. 280)。

主题

- [复制文档](#) (p. 298)
- [添加 Systems Manager 文档 \(控制台\)](#) (p. 298)
- [创建 SSM 文档 \(AWS CLI\)](#) (p. 299)
- [创建 SSM 文档 \(Tools for Windows PowerShell\)](#) (p. 299)

复制文档

在创建文档时，您可以指定 JSON 或 YAML 格式的文档内容。开始创建 SSM 文档最便捷的方式是从一个 Systems Manager 公有文档中复制现有示例。下例展示了如何复制 JSON 示例。

复制 Systems Manager 文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择一个文档。
4. 选择查看详细信息。
5. 选择内容选项卡。
6. 将 JSON 复制到文本编辑器中，并指定自定义文档的详细信息。
7. 以 .json 文件扩展名保存此文档。

在编写完文档内容后，您可以使用以下任一过程将文档添加到 Systems Manager。

- [添加 Systems Manager 文档 \(控制台\)](#) (p. 298)
- [创建 SSM 文档 \(AWS CLI\)](#) (p. 299)
- [创建 SSM 文档 \(Tools for Windows PowerShell\)](#) (p. 299)

添加 Systems Manager 文档 (控制台)

添加 Systems Manager 文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。

2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 选择创建文档。
4. 为文档键入一个描述性名称。
5. 在文档类型列表中，选择您要创建的文档类型。
6. 删除 Content 字段中的方括号，然后将您之前创建的文档粘贴到这里。
7. 选择创建文档以保存文档。

创建 SSM 文档 (AWS CLI)

1. 复制并自定义现有文档，如[复制文档 \(p. 298\)](#)中所述。
2. 使用 AWS CLI 添加文档。

```
aws ssm create-document --content file://path to your file\your file --name "document name" --document-type "Command"
```

Windows 示例

```
aws ssm create-document --content file://c:\temp\PowershellScript.json --name "PowerShellScript" --document-type "Command"
```

Linux 示例

```
aws ssm create-document --content file:///home/ec2-user/RunShellScript.json --name "RunShellScript" --document-type "Command"
```

创建 SSM 文档 (Tools for Windows PowerShell)

1. 复制并自定义现有文档，如[复制文档 \(p. 298\)](#)中所述。
2. 使用 AWS Tools for Windows PowerShell 添加文档。

```
$json = Get-Content C:\your file | Out-String  
New-SSMDocument -DocumentType Command -Name document name -Content $json
```

为 Systems Manager 文档添加标签

可以使用 Systems Manager 控制台、AWS CLI、AWS Tools for Windows 或 [AddTagsToResource](#) API 向 Systems Manager 资源 (包括文档、托管实例、Maintenance Window、Parameter Store 参数和补丁基准) 添加标签。

当您具有相同类型的许多资源时，添加标签会很有用 — 您可以根据分配给资源的标签快速识别特定资源。每个标签都包含您定义的一个键 和一个可选值。

例如，您可以为特定环境、部门、用户、组或时间段的文档添加标签。在为文档添加标签后，您可以通过创建一个指定用户可访问的标签的 IAM 策略来限制对该文档的访问权限。有关使用标签限制对文档的访问权限的更多信息，请参阅[使用标签控制对文档的访问权限 \(p. 301\)](#)。

主题

- [为文档添加标签 \(AWS CLI\) \(p. 300\)](#)
- [为文档添加标签 \(AWS Tools for Windows\) \(p. 300\)](#)
- [为文档添加标签 \(控制台\) \(p. 301\)](#)
- [使用标签控制对文档的访问权限 \(p. 301\)](#)

为文档添加标签 (AWS CLI)

1. 在终端 (Linux, macOS, or Unix) 或命令提示符 (Windows) 中，运行 `list-documents` 命令列出可添加标签的文档。

```
aws ssm list-documents
```

记下您要添加标签的文档的名称。

2. 运行以下命令为文档添加标签。

```
aws ssm add-tags-to-resource --resource-type "Document" --resource-id "document-name"  
--tags "Key=key,Value=value"
```

document-name - 您要添加标签的 Systems Manager 文档的名称。

key 是您提供的自定义密钥的名称。例如，`region` 或 `quarter`。

value 是您要为该密钥提供的值的自定义内容。例如，`west` 或 `Q318`。

如果成功，则命令没有输出。

3. 执行以下命令验证文档标签。

```
aws ssm list-tags-for-resource --resource-type "Document" --resource-id "document-name"
```

为文档添加标签 (AWS Tools for Windows)

1. 打开 AWS Tools for Windows PowerShell 并运行以下命令以列出您可添加标签的文档：

```
Get-SSMDocumentList
```

2. 一次运行一条以下命令来为文档添加标签：

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag  
$tag1.Key = "key"  
$tag1.Value = "value"  
Add-SSMResourceTag -ResourceType "Document" -ResourceId "document-name" -Tag $tag1
```

document-name - 您要添加标签的 Systems Manager 文档的名称。

key 是您提供的自定义密钥的名称。例如，`region` 或 `quarter`。

value 是您要为该密钥提供的值的自定义内容。例如，`west` 或 `Q318`。

如果成功，则命令没有输出。

3. 运行以下命令验证文档标签：

```
Get-SSMResourceTag -ResourceType "Document" -ResourceId "document-name"
```

为文档添加标签 (控制台)

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在左侧导航窗格中，选择文档。
3. 选择现有文档的名称，然后选择标签选项卡。
4. 在第一个框中，为标签输入键，如 `##`。
5. 在第二个框中，为标签输入值，如 `##`。
6. 选择 Save。

使用标签控制对文档的访问权限

在为文档添加标签后，您可以通过创建一个指定用户可访问的标签的 IAM 策略来限制对该文档的访问权限。当用户尝试使用文档时，系统会检查 IAM 策略和为该文档指定的标签。如果用户对分配给该文档的标签没有访问权限，用户会收到访问被拒绝错误。使用以下过程创建一个 IAM 策略来通过使用标签限制对文档的访问权限。

开始前的准备工作

创建文档并添加标签。有关更多信息，请参阅 [为 Systems Manager 文档添加标签 \(p. 299\)](#)。

使用标签限制用户对文档的访问权限

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中选择 Policies，然后选择 Create policy。
3. 选择 JSON 选项卡。
4. 将以下示例策略复制粘贴到此文本字段中，替换示例文本。将 `tag_key` 和 `tag_value` 替换为您的标签的键值对。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetDocument"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/tag_key": [
            "tag_value"
          ]
        }
      }
    }
  ]
}
```

您可以使用以下 Condition 格式在策略中指定多个键。指定多个键会为这些键创建 AND 关系。

```
"Condition": {
```

```
"StringLike":{
  "ssm:resourceTag/tag_key1":[
    "tag_value1"
  ],
  "ssm:resourceTag/tag_key2":[
    "tag_value2"
  ]
}
```

您可以使用以下 Condition 格式在策略中指定多个值。ForAnyValue 为这些值建立 OR 关系。还可以指定 ForAllValues 来建立 AND 关系。

```
"Condition":{
  "ForAnyValue:StringLike":{
    "ssm:resourceTag/tag_key1":[
      "tag_value1",
      "tag_value2"
    ]
  }
}
```

5. 选择查看策略。
6. 在 Name 字段中，指定一个将它标识为目标文档的用户策略的名称。
7. 输入描述。
8. 在摘要部分，验证策略详细信息。
9. 选择 Create policy。
10. 将策略分配给 IAM 用户或组。有关更多信息，请参阅[更改 IAM 用户的权限](#)和[将策略附加到 IAM 组](#)。

将策略附加到 IAM 用户或组账户之后，如果用户尝试使用某个文档，并且用户的策略不允许用户访问该文档的标签 (调用 GetDocument API)，系统将会返回错误。错误类似于以下内容：

"User: **user_name** is not authorized to perform: ssm:GetDocument on resource: **document-name** with the following command."

当文档具有多个标签时，如果用户无权访问这些标签中的任何一个，则用户仍会收到访问被拒绝错误。

共享 Systems Manager 文档

您可以私下或公开共享 Systems Manager 文档。要私下共享文档，请修改文档权限并允许特定个人根据其 Amazon Web Services (AWS) ID 访问文档。要公开共享 Systems Manager 文档，请修改文档权限并指定 All。

Warning

请仅使用从可信来源获取的共享 Systems Manager 文档。使用任何共享文档时，请务必在使用前仔细查看文件内容，了解它会如何更改您的实例配置。有关共享文档最佳实践的更多信息，请参阅 [共享 Systems Manager 文档的共享和使用指南 \(p. 303\)](#)。

限制

在开始使用 Systems Manager 文档时，请注意下列限制。

- 仅所有者可共享文档。
- 您必须先停止共享文档，然后才能删除它。有关更多信息，请参阅 [修改共享文档的权限 \(p. 305\)](#)。
- 您最多可与 1000 个 AWS 账户共享一个文档。要增加此限制，请转到 [AWS 支持中心](#) 并提交限制增加请求表单。

- 您可公开共享最多 5 个 Systems Manager 文档。要增加此限制，请转到 [AWS 支持中心](#) 并提交限制增加请求表单。

有关 Systems Manager 限制的更多信息，请参阅 [AWS Systems Manager 限制](#)。

内容

- [共享 Systems Manager 文档的共享和使用指南 \(p. 303\)](#)
- [共享 Systems Manager 文档 \(p. 303\)](#)
- [修改共享文档的权限 \(p. 305\)](#)
- [使用共享的 Systems Manager 文档 \(p. 306\)](#)

共享 Systems Manager 文档的共享和使用指南

在共享或使用共享文档之前，请阅读以下指南。

删除敏感信息

请仔细审查您的 Systems Manager 文档并删除任何敏感信息。例如，请确保文档不包含您的 AWS 凭证。如果您与特定个人共享文档，这些用户可查看文档中的信息。如果您公开共享文档，则任何人都可查看文档中的信息。

使用 IAM 用户信任策略限制 Run Command 操作

将为有权访问文档的用户创建限制性 AWS Identity and Access Management (IAM) 用户策略。此 IAM 策略确定用户可在 Amazon EC2 控制台中查看或通过使用 AWS CLI 或 AWS Tools for Windows PowerShell 调用 `ListDocuments` 查看的 Systems Manager 文档。该策略还限制用户可使用 Systems Manager 文档执行的操作。您可创建限制性策略，以使用户只能使用特定文档。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

在使用共享文档之前审查其内容

审查与您共享的每个文档 (特别是公开文档) 的内容，以了解将在您的实例上执行的命令。一个文档在运行后可能会有意或无意具有负面影响。如果文档引用外部网络，请在运行文档前审查外部源。

使用文档哈希发送命令

在共享文档时，系统将创建 Sha-256 哈希并将其分配给文档。系统还将保存文档内容的快照。使用共享文档发送命令时，您可在命令中指定哈希以确保下列条件为 true：

- 您正在从正确的 Systems Manager 文档执行命令
- 在与您共享之后文档内容未更改。

如果哈希与指定文档不匹配，或者共享文档的内容已更改，则命令将返回 `InvalidDocument` 异常。请注意：哈希无法验证来自外部位置的文档内容。

共享 Systems Manager 文档

您可以使用、AWS Systems Manager 控制台来共享 Systems Manager 文档；或通过使用 AWS CLI、AWS Tools for Windows PowerShell 或 AWS 软件开发工具包，以编程方式调用 `ModifyDocumentPermission` API 操作来共享 Systems Manager 文档。在共享文档之前，获取要与之共享文档的人的 AWS 账户 ID。您将在共享文档时指定这些账户 ID。

主题

- [共享文档 \(控制台\) \(p. 304\)](#)
- [共享文档 \(AWS CLI\) \(p. 304\)](#)
- [共享文档 \(AWS Tools for Windows PowerShell\) \(p. 305\)](#)

共享文档 (控制台)

共享文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 在文档列表中，选择要共享的文档，然后选择查看详细信息。在权限选项卡中，确保您是文档所有者。只有文档所有者才可共享文档。
4. 选择 Edit。
5. 要公开共享命令，请选择 Public，然后选择 Save。要私下共享命令，请选择私有，输入 AWS 账户 ID，选择添加权限，然后选择保存。

共享文档 (AWS CLI)

以下步骤需要您为您的 CLI 会话指定区域。Run Command 目前在以下 Systems Manager 区域中可用。

1. 在本地计算机上打开 AWS CLI 并执行以下命令来指定凭证。

```
aws config

AWS Access Key ID: [your key]
AWS Secret Access Key: [your key]
Default region name: region
Default output format [None]:
```

##代表 AWS Systems Manager 支持的 AWS 区域的区域标识符，例如 us-east-2 表示 US East (Ohio) Region。有关受支持##值的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager 区域和终端节点表](#) 的区域列。

2. 使用以下命令列出可供您使用的所有 Systems Manager 文档。此列表包括您已创建的文档和与您共享的文档。

```
aws ssm list-documents --document-filter-list key=Owner,value=all
```

3. 使用以下命令获取特定文档。

```
aws ssm get-document --name document name
```

4. 使用以下命令获取文档的描述。

```
aws ssm describe-document --name document name
```

5. 使用以下命令查看文档的权限。

```
aws ssm describe-document-permission --name document name --permission-type Share
```

6. 使用以下命令修改文档的权限并共享文档。您必须是文档的所有者才能编辑权限。此命令与特定个体基于其 AWS 账户 ID 私下共享文档。

```
aws ssm modify-document-permission --name document name --permission-type Share --
account-ids-to-add AWS account ID
```

使用以下命令公开共享文档。

```
aws ssm modify-document-permission --name document name --permission-type Share --  
account-ids-to-add 'all'
```

共享文档 (AWS Tools for Windows PowerShell)

以下步骤需要您为您的 PowerShell 会话指定区域。Run Command 目前在以下 Systems Manager [区域](#)中可用。

1. 在本地计算机上打开 AWS Tools for Windows PowerShell 并执行以下命令来指定凭证。

```
Set-AWSCredentials -AccessKey your key -SecretKey your key
```

2. 使用以下命令为 PowerShell 会话设置区域。此示例使用 us-west-2 区域。

```
Set-DefaultAWSRegion -Region us-west-2
```

3. 使用以下命令列出可供您使用的所有 Systems Manager 文档。此列表包括您已创建的文档与与您共享的文档。

```
Get-SSMDocumentList -DocumentFilterList (@{"key"="Owner";"value"="All"})
```

4. 使用以下命令获取特定文档。

```
Get-SSMDocument -Name document name
```

5. 使用以下命令获取文档的描述。

```
Get-SSMDocumentDescription -Name document name
```

6. 使用以下命令查看文档的权限。

```
Get-SSMDocumentPermission -Name document name -PermissionType Share
```

7. 使用以下命令修改文档的权限并共享文档。您必须是文档的所有者才能编辑权限。此命令与特定个体基于其 AWS 账户 ID 私下共享文档。

```
Edit-SSMDocumentPermission -Name document name -PermissionType Share -  
AccountIdsToAdd AWS account ID
```

使用以下命令公开共享文档。

```
Edit-SSMDocumentPermission -Name document name -AccountIdsToAdd ('all') -PermissionType  
Share
```

修改共享文档的权限

如果您共享一条命令，则在您删除对 Systems Manager 文档的访问权限或删除 Systems Manager 文档之前，用户可查看和使用该命令。但是，只要文档已共享，您就无法删除它。您必须先停止共享，然后再删除它。

主题

- [停止共享文档 \(控制台\) \(p. 306\)](#)
- [停止共享文档 \(AWS CLI\) \(p. 306\)](#)
- [停止共享文档 \(AWS Tools for Windows PowerShell\) \(p. 306\)](#)

停止共享文档 (控制台)

停止共享文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Documents。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后在导航窗格中选择 Documents。

3. 在文档列表中，选择要停止共享的文档，然后选择查看详细信息。在权限选项卡中，确保您是文档所有者。只有文档所有者才可停止共享文档。
4. 选择 Edit。
5. 选择 X 以删除不应再具有此命令的访问权限的 AWS 账户 ID，然后选择保存。

停止共享文档 (AWS CLI)

在本地计算机上打开 AWS CLI，然后执行以下命令停止共享命令。

```
aws ssm modify-document-permission --name document name --permission-type Share --account-ids-to-remove 'AWS account ID'
```

停止共享文档 (AWS Tools for Windows PowerShell)

在本地计算机上打开 AWS Tools for Windows PowerShell，然后执行下列命令停止共享命令。

```
Edit-SSMDocumentPermission -Name document name -AccountIdsToRemove AWS account ID -PermissionType Share
```

使用共享的 Systems Manager 文档

共享 Systems Manager 文档时，系统将生成一个 Amazon 资源名称 (ARN) 并将其分配给命令。如果您从 Amazon EC2 控制台选择并执行共享文档，则不会看到此 ARN。但如果您要从命令行应用程序执行共享 Systems Manager 文档，则必须指定完整的 ARN。当您执行列出文档的命令时，将为您显示 Systems Manager 文档的完整 ARN。

Note

您无需为 AWS 公有文档 (以 AWS-* 开头的文档) 或您拥有的命令指定 ARN。

主题

- [使用共享的 Systems Manager 文档 \(AWS CLI\) \(p. 306\)](#)
- [使用共享的 Systems Manager 文档 \(AWS Tools for Windows PowerShell\) \(p. 307\)](#)

使用共享的 Systems Manager 文档 (AWS CLI)

列出所有公有 Systems Manager 文档

```
aws ssm list-documents --document-filter-list key=Owner,value=Public
```

列出已与您共享的私有 Systems Manager 文档

```
aws ssm list-documents --document-filter-list key=Owner,value=Private
```

列出可供您使用的所有 Systems Manager 文档

```
aws ssm list-documents --document-filter-list key=Owner,value=All
```

使用完整 ARN 通过共享 Systems Manager 文档执行命令

```
aws ssm send-command --document-name FullARN/name
```

例如：

```
aws ssm send-command --document-name arn:aws:ssm:us-east-2:12345678912:document/  
highAvailabilityServerSetup --instance-ids i-12121212
```

使用共享的 Systems Manager 文档 (AWS Tools for Windows PowerShell)

列出所有公有 Systems Manager 文档

```
Get-SSMDocumentList -DocumentFilterList @(New-Object  
Amazon.SimpleSystemsManagement.Model.DocumentFilter("Owner", "Public"))
```

列出已与您共享的私有 Systems Manager 文档

```
Get-SSMDocumentList -DocumentFilterList @(New-Object  
Amazon.SimpleSystemsManagement.Model.DocumentFilter("Owner", "Private"))
```

获取有关已与您共享的 Systems Manager 文档的信息

```
Get-SSMDocument -Name FullARN/name
```

例如：

```
Get-SSMDocument -Name arn:aws:ssm:us-east-2:12345678912:document/  
highAvailabilityServerSetup
```

获取已与您共享的 Systems Manager 文档的描述

```
Get-SSMDocumentDescription -Name FullARN/name
```

例如：

```
Get-SSMDocumentDescription -Name arn:aws:ssm:us-east-2:12345678912:document/  
highAvailabilityServerSetup
```

使用完整 ARN 通过共享 Systems Manager 文档执行命令

```
Send-SSMCommand -DocumentName FullARN/name -InstanceId IDs
```

例如：

```
Send-SSMCommand -DocumentName arn:aws:ssm:us-east-2:555450671542:document/  
highAvailabilityServerSetup -InstanceId @"i-273d4e9e"}
```

创建复合文档

复合 SSM 文档是自定义文档，可通过运行一个或多个次要 SSM 文档执行一系列操作。复合文档允许您为诸如引导软件或加入域的实例等常见任务创建一组标准的 SSM 文档，从而提升了基础设施即代码。您随后可以跨 AWS 账户共享这些文档，从而减少 SSM 文档的维护工作并确保文档一致性。

例如，您可以创建执行以下操作的复合文档：

1. 将 SSM 代理更新到最新版本。
2. 安装已列入白名单的所有补丁。
3. 安装防病毒软件。
4. 从 GitHub 下载脚本并运行这些脚本。

在本例中，您的自定义 SSM 文档包含执行下面这些操作的以下插件：

1. 可运行 AWS-UpdateSSMAgent 文档的 `aws:runDocument` 插件，这会将 SSM 代理更新到最新版本。
2. 可运行 AWS-ApplyPatchBaseline 文档的 `aws:runDocument` 插件，这会安装已列入白名单的所有补丁。
3. 可运行 AWS-InstallApplication 文档的 `aws:runDocument` 插件，这会安装防病毒软件。
4. 可从 GitHub 下载脚本并运行这些脚本的 `aws:downloadContent` 插件。

复合和次要文档可存储在 Systems Manager、GitHub (公有和私有存储库) 或 Amazon S3 中。可创建 JSON 或 YAML 格式的复合文档和辅助文档。

Note

复合文档只能运行三个文档的最大文档深度。这意味着复合文档可以调用子文档；该子文档可调用最后一个文档。

创建复合文档

要创建复合文档，请在自定义 SSM 文档中添加 [aws:runDocument \(p. 328\)](#) 插件并指定必需的输入。下面是可执行以下操作的复合文档示例：

1. 运行 [aws:downloadContent \(p. 324\)](#) 插件可从公有存储库将 SSM 文档下载到名为 bootstrap 的本地目录。SSM 文档称为 StateManagerBootstrap.yml (YAML 文档)。
2. 运行 `aws:runDocument` 插件可运行 StateManagerBootstrap.yml 文档。未指定任何参数。
3. 运行 `aws:runDocument` 插件可运行 AWS-ConfigureDocker 预定义的 SSM 文档。指定的参数会在实例上安装 Docker。

```
{  
  "schemaVersion": "2.2",  
  "description": "My composite document for bootstrapping software and installing Docker.",  
  "parameters": {  
  },  
  "mainSteps": [  
    {  
      "action": "aws:downloadContent",
```

```
    "name": "downloadContent",
    "inputs": {
      "sourceType": "GitHub",
      "sourceInfo": "{\\"owner\\":\\"TestUser1\\",\\"repository\\":\\"TestPublic\\", \\"path\\":\\"documents/bootstrap/StateManagerBootstrap.yml\\"}",
      "destinationPath": "bootstrap"
    }
  },
  {
    "action": "aws:runDocument",
    "name": "runDocument",
    "inputs": {
      "documentType": "LocalPath",
      "documentPath": "bootstrap",
      "documentParameters": "{}"
    }
  },
  {
    "action": "aws:runDocument",
    "name": "configureDocker",
    "inputs": {
      "documentType": "SSMDocument",
      "documentPath": "AWS-ConfigureDocker",
      "documentParameters": "{\\"action\\":\\"Install\\"}"
    }
  }
]
```

相关主题

- 有关在使用 Run Command 调用脚本时重启服务器和实例的信息，请参阅[通过脚本重启托管实例 \(p. 187\)](#)。
- 有关创建 SSM 文档的更多信息，请参阅[创建 Systems Manager 文档 \(p. 297\)](#)。
- 有关可添加到自定义 SSM 文档的插件的更多信息，请参阅[SSM 文档插件参考 \(p. 312\)](#)。
- 如果只想从远程位置运行文档 (无需创建复合文档)，请参阅[从远程位置运行文档 \(p. 309\)](#)。

从远程位置运行文档

您可以使用 AWS-RunDocument 预定义的 SSM 文档从远程位置运行 SSM 文档。此文档目前支持以下远程位置：

- GitHub 存储库 (公有和私有)
- Amazon S3
- 保存在 Systems Manager 中的文档

以下过程介绍如何使用控制台运行远程 SSM 文档。此过程演示如何使用 Run Command 运行远程文档，不过您也可以使用 状态管理器 或 Automation 运行远程文档。

开始前的准备工作

必须先完成以下任务才能运行远程文档。

- 创建 SSM 文档并在远程位置保存该文档。有关更多信息，请参阅 [创建 Systems Manager 文档 \(p. 297\)](#)。
- 如果您打算运行存储在私有 GitHub 存储库中的远程文档，则必须为 GitHub 安全访问令牌创建 Systems Manager SecureString 参数。通过 SSH 手动传递令牌无法访问私有 GitHub 存储库中的远程文档。访

令牌必须作为 Systems Manager SecureString 参数传递。有关创建 SecureString 参数的更多信息，请参阅[创建 Systems Manager 参数 \(p. 369\)](#)。

运行远程文档 (控制台)

运行远程文档

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Run Command。

3. 选择运行命令。
4. 在 Document 列表中，选择 AWS-RunDocument。
5. 在 Targets 部分中，通过手动指定标签或选择实例来标识您要运行此操作的实例。

Note

如果选择手动选择实例，而列表中不包含您预期看到的实例，请参阅[我的实例在哪里？ \(p. 202\)](#)中的故障排除提示。

6. (可选) 在 Rate control 中：
 - 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。
7. 在 Source Type 列表中，选择一个选项。
 - 如果选择 GitHub，则采用以下格式指定源信息信息：

```
{ "owner": "owner_name", "repository": "repository_name", "path": "path_to_document",  
  "tokenInfo": "{{ssm-secure:SecureString_parameter_name}}" }
```

例如：

```
{ "owner": "TestUser1", "repository": "SSMTestDocsRepo", "path": "SSMDocs/  
mySSMdoc.yml", "tokenInfo": "{{ssm-secure:myAccessTokenParam}}" }
```

- 如果选择 S3，则采用以下格式指定源信息信息：

```
{ "path": "URL_to_document_in_S3" }
```

例如：

```
{ "path": "https://s3.amazonaws.com/aws-executecommand-test/scripts/ruby/  
mySSMdoc.json" }
```

- 如果选择 SSMDocument，则采用以下格式指定源信息信息：

```
{"name": "document_name"}
```

例如：

```
{"name": "mySSMdoc"}
```

8. 在文档参数字段中，键入远程 SSM 文档的参数。例如，如果运行 AWS-RunPowerShell 文档，则可指定：

```
{"commands": ["date", "echo \"Hello World\""]}
```

如果运行 AWS-ConfigureAWSPack 文档，则可指定：

```
{  
  "action": "Install",  
  "name": "AWSPVDriver"  
}
```

9. 在 Other parameters 中：

- 在 Comment 框中，键入有关此命令的信息。
- 在 Timeout (seconds) 中，指定整个命令执行失败之前系统要等待的秒数。

10. (可选) 在 Rate control 中：

- 在 Concurrency 中，指定要同时对其运行此命令的实例的数量或百分比。

Note

如果通过选择 Amazon EC2 标签选择了目标，但不确定有多少个实例使用所选标签，则可以通过指定百分比来限制可同时运行此文档的实例的数量。

- 在 Error threshold 中，指定此命令在一定数量或百分比的实例上失败后何时停止在其他实例上运行它。例如，如果您指定 3 个错误，Systems Manager 将在收到第 4 个错误时停止发送命令。仍在处理命令的实例也可能发送错误。

11. 在 Output options 部分中，如果您要将命令输出保存到文件，请选择 Write command output to an Amazon S3 bucket。在框中键入存储桶和前缀 (文件夹) 名称。

Note

授予将数据写入 S3 存储桶的能力的 S3 权限是分配给实例的实例配置文件的权限，而不是执行此任务的 IAM 用户的权限。有关更多信息，请参阅 [配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

12. 在 SNS Notifications 部分中，如果您希望发送有关命令执行状态的通知，请选中 Enable SNS notifications 复选框。

有关为 Run Command 配置 Amazon SNS 通知的更多信息，请参阅 [Run Command 配置 Amazon SNS 通知 \(p. 174\)](#)。

13. 选择 Run。

Note

有关在使用 Run Command 调用脚本时重启服务器和实例的信息，请参阅 [通过脚本重启托管实例 \(p. 187\)](#)。

SSM 文档插件参考

此参考介绍可在 AWS Systems Manager (SSM) 文档中指定的操作或插件。本参考不包含有关 AWS Systems Manager 自动化文档插件的信息。有关自动化文档插件的信息，请参阅[Systems Manager 自动化文档参考 \(p. 334\)](#)。

Systems Manager 通过读取 Systems Manager 文档的内容确定在托管实例上执行的操作。每个文档都包含代码执行部分。根据文档的架构版本，此代码执行部分可能包含一个或多个插件或步骤。为了便于理解本帮助主题，我们将这些插件和步骤都称为插件。本部分包含有关每个 Systems Manager 插件的信息。有关文档的更多信息，包括创建文档以及不同架构版本之间区别的信息，请参阅[AWS Systems Manager 文档 \(p. 289\)](#)。

Note

此处介绍的部分插件仅在 Windows Server 实例或 Linux 实例上运行。应注意每个插件的平台依赖性。

内容

- [顶级元素 \(p. 312\)](#)
- [type 示例 \(p. 313\)](#)
- [aws:applications \(p. 315\)](#)
- [aws:cloudWatch \(p. 316\)](#)
- [aws:configureDocker \(p. 321\)](#)
- [aws:configurePackage \(p. 322\)](#)
- [aws:domainJoin \(p. 323\)](#)
- [aws:downloadContent \(p. 324\)](#)
- [aws:psModule \(p. 325\)](#)
- [aws:refreshAssociation \(p. 326\)](#)
- [aws:runDockerAction \(p. 327\)](#)
- [aws:runDocument \(p. 328\)](#)
- [aws:runPowerShellScript \(p. 329\)](#)
- [aws:runShellScript \(p. 331\)](#)
- [aws:softwareInventory \(p. 331\)](#)
- [aws:updateAgent \(p. 332\)](#)
- [aws:updateSSMAgent \(p. 333\)](#)

顶级元素

顶级元素通常用于所有 Systems Manager 文档。顶级元素提供 Systems Manager 文档的结构。

属性

schemaVersion

架构的版本。

类型：版本

必需：是

description

关于配置的描述。

类型：字符串

必需：否

parameters

`parameters` 是一个结构，其中包含处理文档时要执行的一个或多个参数。您可在运行时指定文档中的参数或者使用 Systems Manager Parameter Store 指定参数。有关更多信息，请参阅 [AWS Systems Manager Parameter Store \(p. 361\)](#)。

类型：结构

`parameters` 结构接受以下字段和值：

- `type`：(必需) 允许的值包括：`String`、`StringList`、`Boolean`、`Integer`、`MapList` 和 `StringMap`。要查看每种类型的示例，请参阅下一节中的 [type 示例 \(p. 313\)](#)。
- `description`：(可选) 关于参数的描述。
- `default`：(可选) 参数的默认值或对 Parameter Store 中参数的引用。
- `allowedValues`：(可选) 参数的允许值。
- `allowedPattern`：(可选) 参数必须匹配的正则表达式。
- `displayType`：(可选) 用于在 AWS 控制台中显示 `textfield` 或 `textarea`。`textfield` 是单行文本框。`textarea` 是多行文本区域。
- `minItems`：(可选) 允许的最小项目数。
- `maxItems`：(可选) 允许的最大项目数。
- `minChars`：(可选) 允许的最小参数字符数。
- `maxChars`：(可选) 允许的最大参数字符数。

runtimeConfig

(仅限 1.2 版架构) 由一个或多个 Systems Manager 插件应用的实例的配置。不保证插件按顺序运行。

类型：Dictionary<string,PluginConfiguration>

必需：否

mainSteps

(仅限 0.3、2.0 和 2.2 版架构) 由一个或多个 Systems Manager 插件应用的实例的配置。插件按照步骤内的操作 进行组织。步骤按文档中列出的先后顺序执行。

类型：Dictionary<string,PluginConfiguration>

必需：否

type 示例

此节包括每个参数 `type` 的示例。

type	描述	示例	示例使用案例
字符串	使用双引号括起来的零个或多个 Unicode 字符序列。使用反斜杠转义。	"i-1234567890abcdef0"	<pre>"InstanceId":{ "type":"String", "description":"(Required) The target EC2 instance ID." }</pre>
StringList	以逗号分隔的 String 项目列表	["cd ~", "pwd"]	<pre>"commands":{ "type":"StringList",</pre>

type	描述	示例	示例使用案例
			<pre>"description": "(Required) Specify a shell script or a command to run.", "minItems": 1, "displayType": "textarea" },</pre>
Boolean	仅接受 true 或 false。不接受 "true" 或 0。	true	<pre>"canRun": { "type": "Boolean", "description": "", "default": true, }</pre>
整数	整数。不接受小数 (如 3.14159) 或使用双引号括起来的数 (如 "3")。	39 或 -5	<pre>"timeout": { "type": "Integer", "description": "The type of action to perform.", "default": 100 }</pre>
StringMap	键到值的映射。键只能为字符串。例如： { "type": "object" }	<pre>{ "NotificationType": "Command", "NotificationEvents": ["Failed"], "NotificationArn": "\$dependency.topicArn" }</pre>	<pre>"notificationConfig" : { "type" : "StringMap", "description" : "The configuration for events to be notified about", "default" : { "NotificationType" : "Command", "NotificationEvents" : ["Failed"], "NotificationArn" : "\$dependency.topicArn" }, "maxChars" : 150 }</pre>

type	描述	示例	示例使用案例
MapList	StringMap 项目的列表。	<pre>[{ "DeviceName" : "/dev/sda1", "Ebs" : { "VolumeSize" : "50" } }, { "DeviceName" : "/dev/sdm", "Ebs" : { "VolumeSize" : "100" } }]</pre>	<pre>"blockDeviceMappings" : { "type" : "MapList", "description" : "The mappings for the create image inputs", "default" : [{"DeviceName":"/ dev/sda1","Ebs": {"VolumeSize":"50"}}, {"DeviceName":"/ dev/sdm","Ebs": {"VolumeSize":"100"}}], "maxItems": 2 }</pre>

aws:applications

在 EC2 实例上安装、修复或卸载应用程序。此插件仅在 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"runtimeConfig":{
  "aws:applications":{
    "properties":[
      {
        "id":"0.aws:applications",
        "action":"{{ action }}",
        "parameters":"{{ parameters }}",
        "source":"{{ source }}",
        "sourceHash":"{{ sourceHash }}"
      }
    ]
  }
}
```

属性

action

要执行的操作。

类型：Enum

有效值：Install | Repair | Uninstall

必需：是

参数

安装程序的参数。

类型：字符串

必需：否

source

应用程序的 .msi 文件的 URL。

类型：字符串

必需：是

sourceHash

.msi 文件的 SHA256 哈希值。

类型：字符串

必需：否

aws:cloudWatch

将数据从 Windows Server 导出到 Amazon CloudWatch 或 Amazon CloudWatch Logs，并使用 CloudWatch 指标监控数据。此插件仅在 Microsoft Windows Server 操作系统中运行。有关配置 CloudWatch 与 Amazon EC2 的集成的更多信息，请参阅[向 Amazon CloudWatch 发送日志、事件和性能计数器](#)。有关文档的更多信息，请参阅[AWS Systems Manager 文档 \(p. 289\)](#)。

您可以导出并监控以下数据类型：

ApplicationEventLog

将应用程序事件日志数据发送到 CloudWatch Logs。

CustomLogs

将任何基于文本的日志文件发送到 CloudWatch 日志。CloudWatch 插件会为日志文件创建指纹。系统随后将数据偏移与每个指纹进行关联。该插件会在出现更改时上传文件，记录偏移，然后将该偏移与指纹关联。此方法用于避免出现这种情况：用户启用插件后，将服务与包含大量文件的目录关联，然后系统会上传所有文件。

Warning

请注意，如果应用程序在轮询期间截断或尝试清除日志，为 `LogDirectoryPath` 指定的任何日志都可能丢失条目。例如，如果您要限制日志文件大小，请在达到此限制时创建新的日志文件，然后继续将数据写入新文件。

ETW

将 Windows 事件跟踪 (ETW) 数据发送到 CloudWatch Logs。不支持 Microsoft Windows Server 2003。

IIS

将 IIS 日志数据发送到 CloudWatch Logs。

PerformanceCounter

将 Windows 性能计数器发送到 CloudWatch。您可以选择不同类别作为指标上传到 CloudWatch。对于要上传的每个性能计数器，创建具有唯一 ID 的 PerformanceCounter 部分 (例如“PerformanceCounter2”、“PerformanceCounter3”等)，然后配置其属性。

Note

如果 SSM 代理或 CloudWatch 插件已停止，则性能计数器数据不再记录到 CloudWatch 中，此行为不同于自定义日志或 Windows 事件日志。自定义日志和 Windows 事件日志会保留性能计数器数据，并在 SSM 代理或 CloudWatch 插件可用时将其上传到 CloudWatch。

SecurityEventLog

将安全事件日志数据发送到 CloudWatch Logs。

SystemEventLog

将系统事件日志数据发送到 CloudWatch Logs。

可为数据定义以下目标：

CloudWatch

发送性能计数器指标数据时所在的目标。可添加具有唯一 ID 的更多部分 (例如，“CloudWatch2”和“CloudWatch3”等)，并为每个新 ID 指定一个不同的区域，将相同数据发送到不同位置。

CloudWatchLogs

发送日志数据时所在的目标。可添加具有唯一 ID 的更多部分 (例如，“CloudWatchLogs2”和“CloudWatchLogs3”等)，并为每个新 ID 指定一个不同的区域，将相同数据发送到不同位置。

语法

```
"runtimeConfig":{
  "aws:cloudWatch":{
    "settings":{
      "startType":"{{ status }}"
    },
    "properties":"{{ properties }}"
  }
}
```

设置和属性

AccessKey

您的访问密钥 ID。如果未使用 IAM 角色启动实例，则必须指定此属性。此属性不能与 SSM 一起使用。

类型：字符串

必需：否

CategoryName

性能监视器提供的性能计数器类别。

类型：字符串

必需：是

CounterName

性能监视器提供的性能计数器名称。

类型：字符串

必需：是

CultureName

记录该时间戳的区域位置。如果 CultureName 为空，则它默认为您的 Windows Server 实例当前所使用的相同区域位置。

类型：字符串

有效值：有关支持的值的列表，请参阅 Microsoft 网站上的[国家语言支持 \(NLS\)](#)。请注意，不支持 div、div-MV、hu 和 hu-HU 值。

必需：否

DimensionName

Amazon CloudWatch 指标的维度。如果您要指定 DimensionName，则必须指定 DimensionValue。这些参数在列出指标时提供另一个视图。您可以对多个指标使用同一个维度，以便查看属于特定维度的所有指标。

类型：字符串

必需：否

DimensionValue

Amazon CloudWatch 指标的维度值。

类型：字符串

必需：否

编码

要使用的文件编码 (例如 UTF-8)。使用编码名称，而不是显示名称。

类型：字符串

有效值：有关支持的值的列表，请参阅 MSDN 库中的[Encoding](#) 类主题。

必需：是

筛选

日志名称的前缀。将此参数留空以监控所有文件。

类型：字符串

有效值：有关支持的值的列表，请参阅 MSDN 库中的[FileSystemWatcherFilter](#) 属性主题。

必需：否

Flows

要上传的每个数据类型以及数据的目标 (CloudWatch 或 CloudWatch Logs)。例如，要将在 "Id": "PerformanceCounter" 下定义的性能计数器发送到在 "Id": "CloudWatch" 下定义的 CloudWatch 目标，请输入 "PerformanceCounter,CloudWatch"。同样，要将自定义日志、ETW 日志和系统日志发送到在 "Id": "ETW" 下定义的 CloudWatch Logs 目标，请输入 "(ETW),CloudWatchLogs"。此外，还可以将相同性能计数器或日志文件发送到多个目标。例如，要将应用程序日志发送到在 "Id": "CloudWatchLogs" 和 "Id": "CloudWatchLogs2" 下定义的两个不同目标，请输入 "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)"。

类型：字符串

有效值 (源)：ApplicationEventLog | CustomLogs | ETW | PerformanceCounter | SystemEventLog | SecurityEventLog

有效值 (目标)：CloudWatch | CloudWatchLogs | CloudWatchⁿ | CloudWatchLogsⁿ

必需：是

FullName

组件的完整名称。

类型：字符串

必需：是

Id

标识数据源或目标。此标识符在配置文件中必须是唯一的。

类型：字符串

必需：是

InstanceName

性能计数器实例的名称。请勿使用星号 (*) 标识所有实例，因为每个性能计数器组件仅支持一个指标。不过可以使用 _Total。

类型：字符串

必需：是

级别

发送到 Amazon CloudWatch 的消息的类型。

类型：字符串

有效值：

- 1 – 仅上传错误消息。
- 2 – 仅上传警告消息。
- 4 – 仅上传信息消息。

请注意，您可以将这些值加在一起以包含多种类型的消息。例如，3 表示将包含错误消息 (1) 和警告消息 (2)。值 7 表示将包含错误消息 (1)、警告消息 (2) 和信息消息 (4)。

必需：是

Note

应将 Windows 安全日志的级别设置为 7。

LineCount

标识日志文件的标头中的行数。例如，IIS 日志文件拥有几乎相同的标头。您可以输入 3，系统会读取日志文件标头的前三行以进行识别。在 IIS 日志文件中，第三行是日期和时间戳，各日志文件的日期和时间戳互不相同。

类型：整数

必需：否

LogDirectoryPath

对于 CustomLogs，这是日志在 Amazon EC2 实例上的存储路径。对于 IIS 日志，这是为单个站点存储 IIS 日志的文件夹 (例如 C:\inetpub\logs\LogFiles\W3SVCⁿ)。对于 IIS 日志，仅支持 W3C 日志格式。不支持 IIS、NCSA 和自定义格式。

类型：字符串

必需：是

LogGroup

日志组的名称。此名称会显示在 CloudWatch 控制台的 Log Groups 屏幕上。

类型：字符串

必需：是

LogName

日志文件的名称。

1. 要查找日志的名称，请在事件查看器的导航窗格中，单击 Applications and Services Logs (应用程序和服务日志)。
2. 在日志列表中，右键单击要上传的日志 (例如“Microsoft>Windows>备份>可操作”)，然后单击创建自定义视图。
3. 在 Create Custom View (创建自定义视图) 对话框中，单击 XML 选项卡。LogName 位于 <Select Path=> 标签中 (例如 Microsoft-Windows-Backup)。将此文本复制到 LogName 参数。

类型：字符串

有效值：Application | Security | System | Microsoft-Windows-WinINet/Analytic

必需：是

LogStream

目标日志流。如果使用默认值 {instance_id}，则将该实例的实例 ID 用作日志流名称。

类型：字符串

有效值：{instance_id} | {hostname} | {ip_address} **<log_stream_name>**

如果输入不存在的日志流名称，则 CloudWatch Logs 会自动创建该名称。您可以使用文字字符串或预定义的变量 ({instance_id}、{hostname}、{ip_address})，或所有这三个变量的组合来定义日志流名称。

此参数中指定的日志流名称会显示在 CloudWatch 控制台中的 Log Groups > Streams for **<YourLogStream>** 屏幕上。

必需：是

MetricName

您希望性能数据在其下显示的 CloudWatch 指标。

Note

不要在名称中使用特殊字符。如果您这样做，指标和相关警报可能无法正常工作。

类型：字符串

必需：是

Namespace

您希望将写入性能计数器数据的指标命名空间。

类型：字符串

必需：是

PollInterval

必须在多少秒之后才能上传新性能计数器和日志数据。

类型：整数

有效值：将其设置为 5 秒或 5 秒以上。建议设置为十五秒 (00:00:15)。

必需：是

区域

要发送日志数据的 AWS 区域。虽然可以将性能计数器发送到与日志数据发送目标不同的区域，但是我们建议您将此参数设置为运行实例的区域。

类型：字符串

有效值：AWS 区域的区域 ID 支持 Systems Manager 和 CloudWatch Logs，例如 `us-east-2`、`eu-west-1` 和 `ap-southeast-1`。有关各个服务支持的 AWS 区域的列表，请参阅 AWS General Reference 中的 [AWS Systems Manager](#) 和 [Amazon CloudWatch Logs](#)。

必需：是

SecretKey

您的秘密访问密钥。如果未使用 IAM 角色启动实例，则必须指定此属性。

类型：字符串

必需：否

startType

在实例上启用或禁用 CloudWatch。

类型：字符串

有效值：`Enabled` | `Disabled`

必需：是

TimestampFormat

要使用的时间戳格式。有关支持的值的列表，请参阅 MSDN 库中的 [自定义日期和时间格式字符串](#)。

类型：字符串

必需：是

TimeZoneKind

在日志时间戳中不包含时区信息时提供时区信息。如果此参数留空且时间戳不包括时区信息，则 CloudWatch Logs 默认为本地时区。如果时间戳已包含时区信息，则忽略此参数。

类型：字符串

有效值：`Local` | `UTC`

必需：否

单位

适当的指标计量单位。

类型：字符串

有效值：`秒` | `微秒` | `毫秒` | `字节` | `千字节` | `兆字节` | `千兆字节` | `太兆字节` | `位` | `千位` | `兆位` | `千兆位` | `太兆位` | `百分比` | `计数` | `字节/秒` | `千字节/秒` | `兆字节/秒` | `千兆字节/秒` | `太兆字节/秒` | `位/秒` | `千位/秒` | `兆位/秒` | `千兆位/秒` | `太兆位/秒` | `计数/秒` | `无`

必需：是

aws:configureDocker

(2.0 版或更高版本架构) 将实例配置为使用容器和 Docker。此插件仅在 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"mainSteps": [
```

```
{
  "action": "aws:configureDocker",
  "name": "ConfigureDocker",
  "inputs": {
    "action": "{{ action }}"
  }
}
```

输入

action

要执行的操作类型。

类型：Enum

有效值：Install | Uninstall

必需：是

aws:configurePackage

(2.0 版或更高版本架构) 安装或卸载 AWS 程序包。可用的程序包包括：AWSPVDriver、AwsEnaNetworkDriver、IntelSriovDriver、AwsVssComponents、AmazonCloudWatchAgent 和 AWSSupport-EC2Rescue。此插件仅在 Linux 和 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"mainSteps": [
  {
    "action": "aws:configurePackage",
    "name": "configurePackage",
    "inputs": {
      "name": "{{ name }}",
      "action": "{{ action }}",
      "version": "{{ version }}"
    }
  }
]
```

输入

name

要安装或卸载的 AWS 程序包的名称。可用的程序包包括：AWSPVDriver、AwsEnaNetworkDriver、IntelSriovDriver、AwsVssComponents 和 AmazonCloudWatchAgent。

类型：字符串

必需：是

action

安装或卸载程序包。

类型：Enum

有效值：Install | Uninstall

必需：是

version

要卸载或安装的特定版本的程序包。如果要安装，系统默认安装最新发布的版本。如果要卸载，系统默认卸载当前安装版本。如果没有找到已安装版本，则会下载最新发布的版本，然后运行卸载操作。

类型：字符串

必需：否

aws:domainJoin

将 Amazon EC2 实例加入域。此插件仅在 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"runtimeConfig":{
  "aws:domainJoin":{
    "properties":{
      "directoryId":"{{ directoryId }}",
      "directoryName":"{{ directoryName }}",
      "directoryOU":"{{ directoryOU }}",
      "dnsIpAddresses":"{{ dnsIpAddresses }}"
    }
  }
}
```

属性

directoryId

目录的 ID。

类型：字符串

必需：是

示例："directoryId": "d-1234567890"

directoryName

域的名称。

类型：字符串

必需：是

示例："directoryName": "example.com"

directoryOU

组织部门 (OU)。

类型：字符串

必需：否

示例："directoryOU": "OU=test,DC=example,DC=com"

dnsIpAddresses

DNS 服务器的 IP 地址。

类型：数组

必需：否

示例："dnsIpAddresses": ["198.51.100.1","198.51.100.2"]

示例

有关示例，请参阅 Amazon EC2 User Guide for Windows Instances 中的[将 Windows Server 实例加入 AWS Directory Service 域](#)。

aws:downloadContent

(2.0 版或更高版本架构) 从远程位置下载 SSM 文档和脚本。Linux 和 Windows Server 操作系统支持此插件。

语法

```
"mainSteps": [
{
  "action": "aws:downloadContent",
  "name": "downloadContent",
  "inputs": {
    "sourceType": "{{ sourceType }}",
    "sourceInfo": "{{ sourceInfo }}",
    "destinationPath": "{{ destinationPath }}"
  }
}
```

输入

sourceType

下载源。Systems Manager 目前支持通过以下类型的源下载脚本和 SSM 文档：GitHub、S3 和 SSMDocument。

类型：字符串

必需：是

sourceInfo

从所需源检索内容所需的信息。

类型：StringMap

必需：是

如果 sourceType 为 GitHub，指定以下内容：

- owner：存储库所有者。
- repository：存储库的名称。
- path：您要下载的文件或目录所在的路径。
- getOptions：从不同的分支或不同提交检索内容的附加选项。此参数采用以下格式：
 - branch：*branch_name*
 - commitID：*commitID*

默认为 head。

- tokenInfo：存储访问令牌信息的 Systems Manager 参数 (SecureString 参数)。

```
Example syntax:
{
  "owner": "TestUser",
  "repository": "GitHubTest",
  "path": "scripts/python/test-script",
  "getOptions": "branch:master",
  "tokenInfo": "{{ssm-secure:secure-string-token}}"
}
```

如果 sourceType 为 S3，指定以下内容：

- path：要从 Amazon S3 下载的文件或目录的 URL。

```
Example syntax:
{
  "path": "https://s3.amazonaws.com/aws-executecommand-test/powershell/
helloPowershell.ps1"
}
```

如果 sourceType 为 SSMDocument，指定以下内容之一：

- name：采用以下格式的文档的名称和版本：name:version。版本为可选项。

```
Example syntax:
{
  "name": "Example-RunPowerShellScript:3"
}
```

- name：采用以下格式的文档的 ARN：
arn:aws:ssm:region:account_id:document/document_name

```
{
  "name": "arn:aws:ssm:us-east-2:3344556677:document/MySharedDoc"
}
```

destinationPath

实例上的可选本地路径，用于下载文件。如果不指定路径，内容将下载到命令 ID 的相对路径。

类型：字符串

必需：否

aws:psModule

在 EC2 实例上安装 PowerShell 模块。此插件仅在 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"runtimeConfig":{
  "aws:psModule":{
    "properties":[
      {
        "id":"0.aws:psModule",
```

```
        "runCommand": "{{ commands }}",
        "source": "{{ source }}",
        "sourceHash": "{{ sourceHash }}",
        "workingDirectory": "{{ workingDirectory }}",
        "timeoutSeconds": "{{ executionTimeout }}"
    }
]
```

属性

runCommand

安装模块之后运行的 PowerShell 命令。

类型：列表或数组

必需：否

source

访问实例上应用程序 .zip 文件的 URL 或本地路径。

类型：字符串

必需：否

sourceHash

.zip 文件的 SHA256 哈希值。

类型：字符串

必需：否

timeoutSeconds

在被视为已失败前命令将运行的时间 (以秒为单位)。

类型：字符串

必需：否

workingDirectory

实例上工作目录的路径。

类型：字符串

必需：否

aws:refreshAssociation

(2.0 版或更高版本架构) 按需刷新 (强制应用) 关联。此操作将根据与目标绑定的所选关联或所有关联的定义更改系统状态。此插件仅在 Linux 和 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"action": "aws:refreshAssociation",
  "name": "refreshAssociation",
  "inputs": {
    "associationIds": "{{ associationIds }}"
  }
```

输入

associationIds

关联 ID 的列表。如果为空，则应用与指定目标绑定的所有关联。

类型：StringList

必需：否

aws:runDockerAction

(2.0 版或更高版本架构) 在容器中运行 Docker 操作。此插件仅在 Linux 和 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"mainSteps": [  
  {  
    "action": "aws:runDockerAction",  
    "name": "RunDockerAction",  
    "inputs": {  
      "action": "{{ action }}",  
      "container": "{{ container }}",  
      "image": "{{ image }}",  
      "memory": "{{ memory }}",  
      "cpuShares": "{{ cpuShares }}",  
      "volume": "{{ volume }}",  
      "cmd": "{{ cmd }}",  
      "env": "{{ env }}",  
      "user": "{{ user }}",  
      "publish": "{{ publish }}"  
    }  
  }  
]
```

输入

action

要执行的操作类型。

类型：字符串

必需：是

container

Docker 容器 ID。

类型：字符串

必需：否

image

Docker 映像名称。

类型：字符串

必需：否

cmd

容器命令。

类型：字符串

必需：否

memory

容器内存限制。

类型：字符串

必需：否

cpuShares

容器 CPU 份额 (相对权重)。

类型：字符串

必需：否

volume

容器卷挂载。

类型：StringList

必需：否

env

容器的环境变量。

类型：字符串

必需：否

user

容器的用户名。

类型：字符串

必需：否

publish

容器已发布端口。

类型：字符串

必需：否

aws:runDocument

(2.0 版或更高版本架构) 执行存储在 Systems Manager 或本地共享存储中的 SSM 文档。您可以将此插件与 [aws:downloadContent](#) (p. 324) 插件配合使用，以将远程位置的 SSM 文档下载到本地共享存储，然后运行该文档。Linux 和 Windows Server 操作系统支持此插件。

语法

```
"mainSteps": [  
  {  
    "action": "aws:runDocument",  
    "name": "runDocument",
```

```
"inputs":{
  "documentType":"{{ documentType }}",
  "documentPath":"{{ documentPath }}",
  "documentParameters":"{{ documentParameters }}"
}
```

输入

documentType

要运行的文档类型。您可以运行本地文档 (LocalPath) 或存储在 Systems Manager (SSMDocument) 中的文档。

类型：字符串

必需：是

documentPath

文档的路径。如果 documentType 是 LocalPath，则指定本地共享存储上文档的路径。如果 documentType 是 SSMDocument，则指定文档的名称。

类型：字符串

必需：否

documentParameters

文档的参数。

类型：StringMap

必需：否

aws:runPowerShellScript

运行 PowerShell 脚本或者指定要运行的脚本的路径。此插件在 Microsoft Windows 和 Linux 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

1.2 SSM 文档的语法

```
"runtimeConfig":{
  "aws:runPowerShellScript":{
    "properties":[
      {
        "id":"0.aws:runPowerShellScript",
        "runCommand":"{{ commands }}",
        "workingDirectory":"{{ workingDirectory }}",
        "timeoutSeconds":"{{ executionTimeout }}"
      }
    ]
  }
}
```

2.2 SSM 文档的语法

```
"mainSteps": [
  {
    "action":"aws:runPowerShellScript",
```

```
    "name": "step name",
    "inputs": {
      "timeoutSeconds": "Timeout in seconds",
      "runCommand": "[Command to execute]"
    }
  }
]
```

下面是 schemaVersion 2.2 的一个示例：

```
{
  "schemaVersion": "2.2",
  "description": "Simple test document using the aws:runPowerShellScript plugin.",
  "parameters": {
    "Salutation": {
      "type": "String",
      "description": "(Optional) This is an optional parameter that will be displayed in the output of the command if specified.",
      "allowedPattern": "[a-zA-Z]",
      "default": "World"
    }
  },
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "DisplaySalutation",
      "inputs": {
        "timeoutSeconds": 60,
        "runCommand": [
          "$salutation = '{{ Salutation }}'",
          "",
          "if ( [String]::IsNullOrEmpty( $salutation ) )",
          "{",
          "  $salutation = 'anonymous'",
          "}",
          "",
          "Write-Host ('Hello {0}' -f $salutation)"
        ]
      }
    }
  ]
}
```

属性

runCommand

指定要运行的命令或实例上现有脚本的路径。

类型：列表或数组

必需：是

timeoutSeconds

在被视为已失败前命令将运行的时间 (以秒为单位)。

类型：字符串

必需：否

workingDirectory

实例上工作目录的路径。

类型：字符串

必需：否

aws:runShellScript

运行 Linux shell 脚本或者指定要运行的脚本的路径。此插件仅在 Linux 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"runtimeConfig":{
  "aws:runShellScript":{
    "properties":[
      {
        "id":"0.aws:runShellScript",
        "runCommand":"{{ commands }}",
        "workingDirectory":"{{ workingDirectory }}",
        "timeoutSeconds":"{{ executionTimeout }}"
      }
    ]
  }
}
```

属性

runCommand

指定要运行的命令或实例上现有脚本的路径。

类型：列表或数组

必需：是

timeoutSeconds

在被视为已失败前命令将运行的时间 (以秒为单位)。

类型：字符串

必需：否

workingDirectory

实例上工作目录的路径。

类型：字符串

必需：否

aws:softwareInventory

(2.0 版或更高版本架构) 从实例中收集应用程序库存、AWS 组件、网络配置、Windows 更新和自定义库存。此插件仅在 Linux 和 Microsoft Windows Server 操作系统中运行。有关更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"mainSteps": [
  {
```

```
    "action": "aws:softwareInventory",
    "name": "collectSoftwareInventoryItems",
    "inputs": {
      "applications": "{{ applications }}",
      "awsComponents": "{{ awsComponents }}",
      "networkConfig": "{{ networkConfig }}",
      "windowsUpdates": "{{ windowsUpdates }}",
      "customInventory": "{{ customInventory }}"
    }
  }
```

输入

applications

收集已安装的应用程序的数据。

类型：字符串

必需：否

awsComponents

收集诸如 amazon-ssm-agent 之类的 AWS 组件的数据。

类型：字符串

必需：否

networkConfig

收集网络配置的数据。

类型：字符串

必需：否

windowsUpdates

收集所有 Windows 更新的数据。

类型：字符串

必需：否

customInventory

收集自定义库存的数据。

类型：字符串

必需：否

aws:updateAgent

将 EC2Config 服务更新到最新版本或指定较旧版本。此插件仅在 Microsoft Windows Server 操作系统中运行。有关 EC2Config 服务的更多信息，请参阅[使用 EC2Config 服务配置 Windows 实例](#)。有关文档的更多信息，请参阅[AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"runtimeConfig": {
  "aws:updateAgent": {
```

```
        "properties": {
          "agentName": "Ec2Config",
          "source": "https://s3.region.amazonaws.com/aws-ssm-region/manifest.json",
          "allowDowngrade": "{{ allowDowngrade }}",
          "targetVersion": "{{ version }}"
        }
      }
    }
```

属性

agentName

EC2Config。这是运行 EC2Config 服务的代理的名称。

类型：字符串

必需：是

allowDowngrade

允许将 EC2Config 服务降级为早期版本。如果设置为 False，则只能将该服务升级为更新的版本 (默认)。如果设置为 True，则指定早期版本。

类型：布尔值

必需：否

source

Systems Manager 复制要安装的 EC2Config 版本的位置。您无法更改此位置。

类型：字符串

必需：是

targetVersion

要安装的特定版本的 EC2Config 服务。如果未指定，服务将更新到最新版本。

类型：字符串

必需：否

aws:updateSSMAgent

将 SSM 代理更新到最新版本或指定较旧版本。此插件在 Linux 和 Windows Server 操作系统中运行。有关更多信息，请参阅 [安装和配置 SSM 代理 \(p. 14\)](#)。有关文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

语法

```
"runtimeConfig": {
  "aws:updateSsmAgent": {
    "properties": [
      {
        "agentName": "amazon-ssm-agent",
        "source": "https://s3.region.amazonaws.com/aws-ssm-region/manifest.json",
        "allowDowngrade": "{{ allowDowngrade }}",
        "targetVersion": "{{ version }}"
      }
    ]
  }
}
```

```
}  
}
```

属性

agentName

amazon-ssm-agent。这是在实例上处理请求并运行命令的 Systems Manager 代理的名称。

类型：字符串

必需：是

allowDowngrade

允许将 SSM 代理降级为早期版本。如果设置为 False，则只能将该代理升级为更新的版本 (默认)。如果设置为 True，则指定早期版本。

类型：布尔值

必需：否

source

Systems Manager 复制要安装的 SSM 代理版本的位置。您无法更改此位置。

类型：字符串

必需：是

targetVersion

要安装的特定版本的 SSM 代理。如果未指定，代理将更新到最新版本。

类型：字符串

必需：否

Systems Manager 自动化文档参考

此参考介绍可在 AWS Systems Manager 自动化文档中指定的操作 (或插件)。有关其他 SSM 文档类型插件的信息，请参阅[SSM 文档插件参考 \(p. 312\)](#)。

Systems Manager Automation 会运行自动化文档中定义的步骤。每步都与特定操作相关。此操作确定本步的输入、行为和输出。步骤定义在 Automation 文档的 `mainSteps` 部分。

您无需指定操作或步骤的输出。由与本步关联的操作预先确定输出。当您在 Automation 文档中指定步骤输入时，可以引用前面某步中的一个或多个输出。例如，您可以使 `aws:runInstances` 的输出可用于后续的 `aws:runCommand` 操作。还可以在 Automation 文档的 `Output` 部分引用前面步骤中的输出。

主题

- [所有操作中的常见属性 \(p. 335\)](#)
- [aws:approve \(p. 338\)](#)
- [aws:changeInstanceState \(p. 340\)](#)
- [aws:copyImage \(p. 341\)](#)
- [aws:createImage \(p. 343\)](#)
- [aws:createStack \(p. 344\)](#)
- [aws:createTags \(p. 349\)](#)
- [aws:deleteImage \(p. 350\)](#)

- [aws:deleteStack](#) (p. 350)
- [aws:executeAutomation](#) (p. 351)
- [aws:executeStateMachine](#) (p. 353)
- [aws:invokeLambdaFunction](#) (p. 353)
- [aws:pause](#) (p. 355)
- [aws:runCommand](#) (p. 355)
- [aws:runInstances](#) (p. 357)
- [aws:sleep](#) (p. 360)

所有操作中的常见属性

以下属性是所有操作的常见属性：

```
"mainSteps": [  
  {  
    "name": "name",  
    "action": "action",  
    "maxAttempts": value,  
    "timeoutSeconds": value,  
    "onFailure": "value",  
    "inputs": {  
      ...  
    }  
  },  
  {  
    "name": "name",  
    "action": "action",  
    "maxAttempts": value,  
    "timeoutSeconds": value,  
    "onFailure": "value",  
    "inputs": {  
      ...  
    }  
  }  
]
```

name

在文档的所有步骤名称中必须唯一的标识符。

类型：字符串

必需：是

action

步骤要执行的操作的名称。[aws:runCommand](#) (p. 355)是可在此处指定的操作的示例。本文档提供有关所有可用操作的详细信息。

类型：字符串

必需：是

maxAttempts

在发生故障的情况下应重试步骤的次数。如果值大于 1，则直到所有重试尝试失败后，才会将此步骤视为失败。默认值是 1。

类型：整数

必需：否

timeoutSeconds

步骤的执行超时值。如果超时并且 maxAttempts 的值大于 1，则本步未考虑超时，直至已尝试所有重试。此字段没有默认值。

类型：整数

必需：否

onFailure

指示工作流程在失败时是应中止、继续还是转到其他步骤。该选项的默认值为“中止”。

类型：字符串

有效值：Abort | Continue | step:*step_name*

必需：否

isEnd

此选项在特定步骤结束时停止自动化执行。如果步骤执行失败，自动化执行将停止，否则将成功。默认值为 False。

类型：布尔值

有效值：true | false

必需：否

下面是在文档的 mainSteps 部分中输入此选项的示例：

```
"mainSteps":[
  {
    "name":"InstallMsiPackage",
    "action":"aws:runCommand",
    "onFailure":"step:PostFailure",
    "maxAttempts":2,
    "inputs":{
      "InstanceIds":[
        {
          "i-1234567890abcdef0,i-0598c7d356eba48d7"
        }
      ],
      "DocumentName":"AWS-RunPowerShellScript",
      "Parameters":{
        "commands":[
          "msiexec /i {{packageName}}"
        ]
      }
    },
    "nextStep":"TestPackage"
  },
  {
    "name":"TestPackage",
    "action":"aws:invokeLambdaFunction",
    "maxAttempts":1,
    "timeoutSeconds":500,
    "inputs":{
      "FunctionName":"TestLambdaFunction"
    },
  },
]
```

```
    "isEnd":true
  }
```

nextStep

指定在成功完成自动化工作流程中的一个步骤后，接下来处理哪个步骤。

下面是如何在文档的 `mainSteps` 部分中输入此选项的示例：

```
"mainSteps":[
  {
    "name":"InstallMsiPackage",
    "action":"aws:runCommand",
    "onFailure":"step:PostFailure",
    "maxAttempts":2,
    "inputs":{"
      "InstanceIds":[
        {
          "i-1234567890abcdef0,i-0598c7d356eba48d7"
        }
      ]
    },
    "DocumentName":"AWS-RunPowerShellScript",
    "Parameters":{"
      "commands":["
        msexec /i {{packageName}}"
      ]
    }
  },
  "nextStep":"TestPackage"
}
```

isCritical

将一个步骤指定为成功完成自动化的关键步骤。如果具有此分派的步骤失败，则自动化会将自动化的最终状态报告为失败。该选项的默认值为 `true`。

类型：布尔值

有效值：`true` | `false`

必需：否

下面是如何在文档的 `mainSteps` 部分中输入此选项的示例：

```
{
  "name":"InstallMsiPackage",
  "action":"aws:runCommand",
  "onFailure":"step:SomeOtherStep",
  "isCritical":false,
  "maxAttempts":2,
  "inputs":{"
    "InstanceIds":[
      {
        {
          "i-1234567890abcdef0,i-0598c7d356eba48d7"
        }
      ]
    },
    "DocumentName":"AWS-RunPowerShellScript",
    "Parameters":{"
      "commands":["
```

```
        "msiexec /i {{packageName}}"
    ]
  }
},
"nextStep": "TestPackage"
}
```

inputs

特定于操作的属性。

类型：映射

必需：是

aws:approve

临时暂停 Automation 执行，直至指定委托人批准或拒绝操作。在达到所需批准数后，Automation 执行将恢复。您可在 Automation 文档的 mainSteps 部分中的任何位置插入批准步骤。

在以下示例中，aws:approve 操作临时暂停 Automation 工作流程，直至一个审批者接受或拒绝工作流程。批准后，此文档将运行简单的 PowerShell 命令。

```
{
  "description": "RunInstancesDemo1",
  "schemaVersion": "0.3",
  "assumeRole": "{ assumeRole }",
  "parameters": {
    "assumeRole": {
      "type": "String"
    },
    "message": {
      "type": "String"
    }
  },
  "mainSteps": [
    {
      "name": "approve",
      "action": "aws:approve",
      "timeoutSeconds": 1000,
      "onFailure": "Abort",
      "inputs": {
        "NotificationArn": "arn:aws:sns:us-east-2:12345678901:AutomationApproval",
        "Message": "{ message }",
        "MinRequiredApprovals": 1,
        "Approvers": [
          "arn:aws:iam::12345678901:user/AWS-User-1"
        ]
      }
    },
    {
      "name": "run",
      "action": "aws:runCommand",
      "inputs": {
        "InstanceIds": [
          "i-1a2b3c4d5e6f7g"
        ],
        "DocumentName": "AWS-RunPowerShellScript",
        "Parameters": {
          "commands": [
            "date"
          ]
        }
      }
    }
  ]
}
```

```
}
    }
  }
}
```

您可以在控制台中批准或拒绝等待批准的自动化。

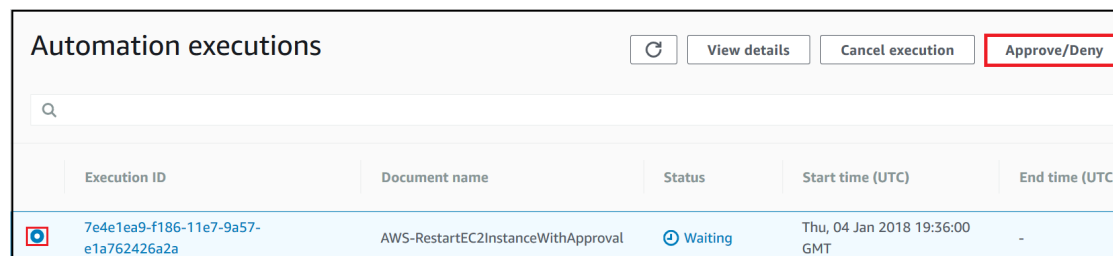
批准或拒绝正在等待的自动化

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Automation。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Automation。

3. 选择状态为正在等待的自动化旁边的选项。



4. 选择批准/拒绝。
5. 查看自动化的详细信息。
6. 选择批准或拒绝，键入评论 (可选)，然后选择提交。

输入

```
{
  "NotificationArn": "arn:aws:sns:us-west-1:12345678901:Automation-ApprovalRequest",
  "Message": "Please approve this step of the Automation.",
  "MinRequiredApprovals": 3,
  "Approvers": [
    "IamUser1",
    "IamUser2",
    "arn:aws:iam::12345678901:user/IamUser3",
    "arn:aws:iam::12345678901:role/IamRole"
  ]
}
```

NotificationArn

Automation 批准的 Amazon SNS 主题的 ARN。当您在 Automation 文档中指定 `aws:approve` 步骤时，Automation 将向此主题发送消息，以让委托人知道必须批准或拒绝 Automation 步骤。Amazon SNS 主题的标题必须使用前缀“Automation”。

类型：字符串

必需：否

Message

发送批准请求时要包含在 SNS 主题中的信息。最大消息长度为 4096 个字符。

类型：字符串

必需：否

MinRequiredApprovals

恢复 Automation 执行所需的批准的最大数。如果您未指定值，系统将默认为 1。此参数的值必须为正数。此参数的值不能超过 Approvers 参数定义的审批者数。

类型：整数

必需：否

Approvers

能够批准或拒绝操作的经 AWS 身份验证的委托人的列表。最大审批者数为 10。您可使用以下任意格式指定委托人：

- AWS Identity and Access Management (IAM) 用户名
- IAM 用户 ARN
- IAM 角色 ARN
- IAM 代入角色用户 ARN

类型：StringList

必需：是

输出

ApprovalStatus

步骤的批准状态。状态可以为下列状态之一：已批准、已拒绝或正在等待。“正在等待”意味着 Automation 正在等待来自审批者的输入。

类型：字符串

ApproverDecisions

包括每位审批者的批准决定的 JSON 映射。

类型：MapList

aws:changeInstanceState

更改或断言实例的状态。

此操作可在断言模式下使用 (不要执行 API 来更改状态，而应验证实例是否处于预期状态。)要使用断言模式，请将 CheckStateOnly 参数设置为 true。当在 Windows 上运行 Sysprep 命令时，此模式很有用。该命令是一种可在后台长时间运行的异步命令。您可以确保在创建 AMI 之前停止实例。

输入

```
{
  "name": "stopMyInstance",
  "action": "aws:changeInstanceState",
  "maxAttempts": 3,
  "timeoutSeconds": 3600,
  "onFailure": "Abort",
  "inputs": {
    "InstanceIds": ["i-1234567890abcdef0"],
    "CheckStateOnly": true,
    "DesiredState": "stopped"
  }
}
```

```
}  
}
```

InstanceIds

实例的 ID。

类型：StringList

必需：是

CheckStateOnly

如果为 false，请将实例状态设置为预期状态。如果为 true，请使用轮询断言预期状态。

类型：布尔值

必需：否

DesiredState

预期状态。

类型：字符串

有效值：running | stopped | terminated

必需：是

Force

如果设置此项，则强制停止实例。则该实例没有机会来刷新文件系统缓存或文件系统元数据。如果您使用此选项，则必须执行文件系统检查和修复流程。我们不建议将该选项用于 Windows 实例。

类型：布尔值

必需：否

AdditionalInfo

预留。

类型：字符串

必需：否

输出

无

aws:copyImage

将 AMI 从任何区域复制到当前区域中。此操作还可以对新的 AMI 进行加密。

输入

此操作支持大多数 CopyImage 参数。有关更多信息，请参阅 [CopyImage](#)。

以下示例在首尔地区创建 AMI 的副本 (SourceImageID：ami-0fe10819。SourceRegion: ap-northeast-2)。新的 AMI 将复制到您启动 Automation 操作的区域。将对复制的 AMI 进行加密，因为可选 Encrypted 标记将设置为 true。

```
{
```

```
"name": "createEncryptedCopy",
"action": "aws:copyImage",
"maxAttempts": 3,
"onFailure": "Abort",
"inputs": {
  "SourceImageId": "ami-0fe10819",
  "SourceRegion": "ap-northeast-2",
  "ImageName": "Encrypted Copy of LAMP base AMI in ap-northeast-2",
  "Encrypted": true
}
```

SourceRegion

源 AMI 当前所在的区域。

类型：字符串

必需：是

SourceImageId

要从源区域复制的 AMI ID。

类型：字符串

必需：是

ImageName

新映像的名称。

类型：字符串

必需：是

ImageDescription

目标映像的描述。

类型：字符串

必需：否

加密

对目标 AMI 进行加密。

类型：布尔值

必需：否

KmsKeyId

在复制操作期间对映像快照进行加密时要使用的 AWS Key Management Service CMK 的完整 Amazon 资源名称 (ARN)。有关更多信息，请参阅 [CopyImage](#)。

类型：字符串

必需：否

ClientToken

您为确保请求幂等性而提供的唯一、区分大小写的标识符。有关更多信息，请参阅 [CopyImage](#)。

类型：字符串

必需：否

输出

ImageId

已复制映像的 ID。

ImageState

已复制映像的状态。

有效值：available | pending | failed

aws:createImage

从正在运行的或已停止的实例创建新的 AMI。

输入

此操作支持大多数 CreateImage 参数。有关更多信息，请参阅 [CreateImage](#)。

```
{
  "name": "createMyImage",
  "action": "aws:createImage",
  "maxAttempts": 3,
  "onFailure": "Abort",
  "inputs": {
    "InstanceId": "i-1234567890abcdef0",
    "ImageName": "AMI Created on{{global:DATE_TIME}}",
    "NoReboot": true,
    "ImageDescription": "My newly created AMI"
  }
}
```

实例 ID

实例的 ID。

类型：字符串

必需：是

ImageName

映像的名称。

类型：字符串

必需：是

ImageDescription

映像的描述。

类型：字符串

必需：否

NoReboot

一种布尔文本。

默认情况下，Amazon EC2 会尝试关闭并重新启动实例，然后再创建映像。如果 `No Reboot (###)` 选项设置为 `true`，则 Amazon EC2 在创建映像前不会关闭实例。如果使用此选项，则无法保证所创建映像上的文件系统的完整性。

如果您希望在从实例创建 AMI 映像后，该实例不运行，请先使用 [aws:changeInstanceState \(p. 340\)](#) 插件停止实例，然后在 `NoReboot` 选项设置为 `true` 的情况下使用此 `aws:createImage` 插件。

类型：布尔值

必需：否

BlockDeviceMappings

适用于实例的块储存设备。

类型：映射

必需：否

输出

ImageId

新建映像的 ID。

ImageState

作为字符串文本值提供的执行脚本。如果输入文本值，则必须为 Base64 编码。

必需：否

aws:createStack

从模板创建新 AWS CloudFormation 堆栈。

输入

```
{
  "name": "makeStack",
  "action": "aws:createStack",
  "maxAttempts": 1,
  "onFailure": "Abort",
  "inputs": {
    "Capabilities": [
      "CAPABILITY_IAM"
    ],
    "StackName": "myStack",
    "TemplateURL": "http://s3.amazonaws.com/mybucket/myStackTemplate",
    "TimeoutInMinutes": 5
  }
}
```

功能

必须在 AWS CloudFormation 可以创建某些堆栈之前指定的值列表。一些堆栈模板中包含的资源会影响您的 AWS 账户中的权限。例如，创新新 AWS Identity and Access Management (IAM) 用户会影响您账户中的权限。对于这些堆栈，您必须通过指定此参数来明确确认它们的功能。

有效值仅为 `CAPABILITY_IAM` 和 `CAPABILITY_NAMED_IAM`。以下资源要求您指定此参数。

- [AWS::IAM::AccessKey](#)

- [AWS::IAM::Group](#)
- [AWS::IAM::InstanceProfile](#)
- [AWS::IAM::Policy](#)
- [AWS::IAM::Role](#)
- [AWS::IAM::User](#)
- [AWS::IAM::UserToGroupAddition](#)

如果您的堆栈模板包含这些资源，我们建议您查看与之关联的所有权限并在必要时编辑其权限。

如果包含 IAM 资源，您可以指定任意一个功能。如果包含具有自定义名称的 IAM 资源，则必须指定 `CAPABILITY_NAMED_IAM`。如果您不指定此参数，则此操作会返回 `InsufficientCapabilities` 错误。

有关更多信息，请参阅[确认 AWS CloudFormation 模板中的 IAM 资源](#)。

类型：字符串的数组

有效值：`CAPABILITY_IAM` | `CAPABILITY_NAMED_IAM`

必需：否

DisableRollback

如果堆栈创建失败，请设置为 `true` 以禁用堆栈回滚。

Conditional：您可以指定 `DisableRollback` 参数或 `OnFailure` 参数，但不能同时指定。

默认值：`false`

类型：布尔值

必需：否

NotificationARNs

用于发布堆栈相关事件的 Amazon SNS 主题 ARN。您可以使用 Amazon SNS 控制台 <https://console.aws.amazon.com/sns/v2/home> 找到 SNS 主题 ARN。

类型：字符串的数组

数组成员：最多 5 项。

必需：否

OnFailure

如果堆栈创建失败，确定要执行的操作。您必须指定 `DO_NOTHING`、`ROLLBACK` 或 `DELETE`。

Conditional：您可以指定 `OnFailure` 参数或 `DisableRollback` 参数，但不能同时指定。

默认值：`ROLLBACK`

类型：字符串

有效值：`DO_NOTHING` | `ROLLBACK` | `DELETE`

必需：否

参数

指定堆栈输入参数的 `Parameter` 结构列表。有关更多信息，请参阅[参数数据类型](#)。

类型：[参数](#)对象数组

必需：否

ResourceTypes

您有权用于此创建堆栈操作的模板资源类型。例如，`AWS::EC2::Instance`、`AWS::EC2::*` 或 `Custom::MyCustomInstance`。使用以下语法描述模板资源类型。

- 对于所有 AWS 资源：

```
AWS::*
```

- 对于所有自定义资源：

```
Custom::*
```

- 对于指定自定义资源：

```
Custom::logical_ID
```

- 对于特定 AWS 服务中的所有资源：

```
AWS::service_name::*
```

- 对于指定 AWS 资源：

```
AWS::service_name::resource_logical_ID
```

如果资源类型列表不包括您创建的资源，那么堆栈创建将会失败。默认情况下，AWS CloudFormation 对所有资源类型授予权限。IAM 将此参数用于 IAM 策略中 AWS CloudFormation 特定的条件密钥。有关更多信息，请参阅[使用 AWS Identity and Access Management 控制访问权限](#)。

类型：字符串的数组

长度约束：最小长度为 1。长度上限为 256。

必需：否

RoleARN

AWS CloudFormation 创建堆栈所代入的 IAM 角色的 Amazon 资源名称 (ARN)。AWS CloudFormation 使用该角色的凭证以您的名义进行调用。AWS CloudFormation 针对堆栈的所有未来操作始终使用此角色。只要用户有权对堆栈进行操作，AWS CloudFormation 会使用此角色，即使用户无权传递它。确保该角色授予最少的权限。

如果您不指定值，则 AWS CloudFormation 会使用之前与堆栈关联的角色。如果角色不可用，则 AWS CloudFormation 会使用您的用户凭证生成的一个临时会话。

类型：字符串

长度约束：最小长度为 20。长度上限为 2048。

必需：否

StackName

与堆栈关联的名称。名称在您创建堆栈的区域中必须是唯一的。

Note

堆栈名称只能包含字母数字字符（区分大小写）和连字符。该名称必须以字母字符开头，且不得超过 128 个字符。

类型：字符串

必需：是

StackPolicyBody

包含堆栈策略正文的结构。有关更多信息，请参阅[防止更新堆栈资源](#)。

Conditional：您可以指定 StackPolicyBody 参数或 StackPolicyURL 参数，但不能同时指定。

类型：字符串

长度约束：最小长度为 1。长度上限为 16384。

必需：否

StackPolicyURL

包含堆栈策略的文件的位置。URL 指向的策略必须位于与堆栈处于同一区域的 Amazon S3 存储桶中。堆栈策略允许的最大文件大小为 16 KB。

Conditional：您可以指定 StackPolicyBody 参数或 StackPolicyURL 参数，但不能同时指定。

类型：字符串

长度约束：最小长度为 1。长度上限为 1350。

必需：否

Tags

与此堆栈关联的键值对。AWS CloudFormation 还可以将这些标签传播到堆栈中创建的资源。您可以指定最多 10 个标签。

类型：标签对象数组

必需：否

TemplateBody

包含最小长度为 1 字节、最大长度为 51200 字节的模板正文的结构。有关更多信息，请参阅[模板剖析](#)。

Conditional：您可以指定 TemplateBody 参数或 TemplateURL 参数，但不能同时指定。

类型：字符串

长度约束：最小长度为 1。

必需：否

TemplateURL

包含模板正文的文件的位置。URL 必须指向一个位于 Amazon S3 存储桶中的模板。模板允许的最大大小为 460800 字节。有关更多信息，请参阅[模板剖析](#)。

Conditional：您可以指定 TemplateBody 参数或 TemplateURL 参数，但不能同时指定。

类型：字符串

长度约束：最小长度为 1。长度上限为 1024。

必需：否

TimeoutInMinutes

堆栈状态变为 CREATE_FAILED 前允许经过的时间。如果未设置 DisableRollback 或将其设置为 false，堆栈将被回滚。

类型：整数

有效范围：最小值为 1。

必需：否

输出

堆栈 ID

堆栈的唯一标识符。

类型：字符串

StackStatus

堆栈的当前状态。

类型：字符串

有效值：CREATE_IN_PROGRESS | CREATE_FAILED | CREATE_COMPLETE
| ROLLBACK_IN_PROGRESS | ROLLBACK_FAILED | ROLLBACK_COMPLETE
| DELETE_IN_PROGRESS | DELETE_FAILED | DELETE_COMPLETE |
UPDATE_IN_PROGRESS | UPDATE_COMPLETE_CLEANUP_IN_PROGRESS |
UPDATE_COMPLETE | UPDATE_ROLLBACK_IN_PROGRESS | UPDATE_ROLLBACK_FAILED |
UPDATE_ROLLBACK_COMPLETE_CLEANUP_IN_PROGRESS | UPDATE_ROLLBACK_COMPLETE |
REVIEW_IN_PROGRESS

必需：是

StackStatusReason

与堆栈状态相关联的成功或失败消息。

类型：字符串

必需：否

有关更多信息，请参阅 [CreateStack](#)。

安全考虑因素

您必须将以下策略分配给 IAM Automation 代入角色，才可以使用 `aws:createStack` 操作。有关代入角色的更多信息，请参阅 [任务 1：为 Automation 创建服务角色 \(p. 110\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:*",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStacks"
      ],
      "Resource": "*"
    }
  ]
}
```

aws:createTags

为 Amazon EC2 实例或 Systems Manager 托管实例创建新标签。

输入

此操作支持大多数 EC2 CreateTags 和 SSM AddTagsToResource 参数。有关更多信息，请参阅 [CreateTags](#) 和 [AddTagsToResource](#)。

以下示例说明如何为 AMI 和实例添加标签以作为特定部门的生产资源。

```
{
  "name": "createTags",
  "action": "aws:createTags",
  "maxAttempts": 3,
  "onFailure": "Abort",
  "inputs": {
    "ResourceType": "EC2",
    "ResourceIds": [
      "ami-9a3768fa",
      "i-02951acd5111a8169"
    ],
    "Tags": [
      {
        "Key": "production",
        "Value": ""
      },
      {
        "Key": "department",
        "Value": "devops"
      }
    ]
  }
}
```

ResourceIds

要为其添加标签的资源的 ID。如果资源类型不是“EC2”，则此字段只能包含单个项目。

类型：字符串列表

必需：是

Tags

要与资源关联的标签。

类型：映射列表

必需：是

ResourceType

要为其添加标签的资源的类型。如果未提供，则使用默认值“EC2”。

类型：字符串

必需：否

有效值: EC2 | ManagedInstance | MaintenanceWindow | Parameter

输出

无

aws:deleteImage

删除指定映像和所有的相关快照。

输入

此操作仅支持一个参数。有关更多信息，请参阅 [DeregisterImage](#) 和 [DeleteSnapshot](#) 的相关文档。

```
{
  "name": "deleteMyImage",
  "action": "aws:deleteImage",
  "maxAttempts": 3,
  "timeoutSeconds": 180,
  "onFailure": "Abort",
  "inputs": {
    "ImageId": "ami-12345678"
  }
}
```

ImageId

要删除的映像的 ID。

类型：字符串

必需：是

输出

无

aws:deleteStack

删除 AWS CloudFormation 堆栈。

输入

```
{
  "name": "deleteStack",
  "action": "aws:deleteStack",
  "maxAttempts": 1,
  "onFailure": "Abort",
  "inputs": {
    "StackName": "${stackName}"
  }
}
```

ClientRequestToken

此 DeleteStack 请求的唯一标识符。如果您计划重试请求以便 AWS CloudFormation 知道您未在尝试删除同名堆栈，请指定此令牌。您可以重试 DeleteStack 请求以验证 AWS CloudFormation 是否收到了它们。

类型：字符串

长度约束：最小长度为 1。长度上限为 128。

模式：[a-zA-Z][-a-zA-Z0-9]*

必需：否

RetainResources.member.N

此输入仅适用于处于 `DELETE_FAILED` 状态的堆栈。您想要保留的资源的逻辑资源 ID 的列表。在删除时，AWS CloudFormation 删除堆栈，但不删除保留资源。

如果无法删除某个资源 (例如非空 Amazon S3 存储桶)，但需要删除堆栈，则保留资源会很有用。

类型：字符串的数组

必需：否

RoleARN

AWS CloudFormation 创建堆栈所代入的 IAM 角色的 Amazon 资源名称 (ARN)。AWS CloudFormation 使用该角色的凭证以您的名义进行调用。AWS CloudFormation 针对堆栈的所有未来操作始终使用此角色。只要用户有权对堆栈进行操作，AWS CloudFormation 会使用此角色，即使用户无权传递它。确保该角色授予最少的权限。

如果您不指定值，则 AWS CloudFormation 会使用之前与堆栈关联的角色。如果角色不可用，则 AWS CloudFormation 会使用您的用户凭证生成的一个临时会话。

类型：字符串

长度约束：最小长度为 20。长度上限为 2048。

必需：否

StackName

与堆栈关联的名称或唯一堆栈 ID。

类型：字符串

必需：是

安全考虑因素

您必须将以下策略分配给 IAM Automation 代入角色，才可以使用 `aws:deleteStack` 操作。有关代入角色的更多信息，请参阅[任务 1：为 Automation 创建服务角色 \(p. 110\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:*",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStacks"
      ],
      "Resource": "*"
    }
  ]
}
```

aws:executeAutomation

通过调用辅助自动化文件执行辅助自动化工作流程。借助此操作，您可以为最常见的工作流程创建自动化文档，并在自动化执行期间引用这些文档。因为无需跨类似文档复制步骤，此操作可以简化您的自动化文档。

辅助自动化将在启动初始自动化的用户环境中运行。这意味着辅助自动化将使用与启动初次自动化的用户相同的 IAM 角色和用户账户。

Important

如果您在使用代入角色 (使用 iam:passRole 策略的角色) 的辅助自动化中指定参数，则启动主要自动化的用户或角色必须具有在辅助自动化中传递指定的代入角色的权限。有关为自动化设置代入角色的更多信息，请参阅[方法 2：使用 IAM 为 Automation 配置角色 \(p. 110\)](#)。

输入

```
{
  "name": "Secondary_Automation_Workflow",
  "action": "aws:executeAutomation",
  "maxAttempts": 3,
  "timeoutSeconds": 3600,
  "onFailure": "Abort",
  "inputs": {
    "DocumentName": "secondaryWorkflow",
    "RuntimeParameters": {
      "instanceIds": [
        "i-1234567890abcdef0"
      ]
    }
  }
}
```

DocumentName

要在步骤中执行的辅助自动化文档的名称。该文档必须属于与主要自动化文档相同的 AWS 账户。

类型：字符串

必需：是

DocumentVersion

要在步骤中执行的辅助自动化文档的版本。如果未指定，自动化将运行默认文档版本。

类型：字符串

必需：是

RuntimeParameters

辅助文档执行所需的参数。映射使用以下格式：{"parameter1": ["value1"], "parameter2": ["value2"]}

类型：映射

必需：否

输出

输出

辅助执行生成的输出。您可以使用以下格式引用输出：**Secondary_Automation_Step_Name**.Output

类型：StringList

ExecutionId

辅助执行的执行 ID。

类型：字符串

Status

辅助执行的状态。

类型：字符串

aws:executeStateMachine

执行 AWS Step Functions 状态机。

输入

此操作支持 Step Functions [StartExecution](#) API 操作的大多数参数。

```
{
  "name": "executeTheStateMachine",
  "action": "aws:executeStateMachine",
  "inputs": {
    "stateMachineArn": "StateMachine_ARN",
    "input": "{\"parameters\": \"values\"}",
    "name": "name"
  }
}
```

stateMachineArn

Step Functions 状态机的 ARN。

类型：字符串

必需：是

name

执行的名称。

类型：字符串

必需：否

input

包含执行的 JSON 输入数据的字符串。

类型：字符串

必需：否

aws:invokeLambdaFunction

调用指定的 Lambda 函数。

输入

此操作支持 Lambda 服务的大多数调用参数。有关更多信息，请参阅[调用](#)。

```
{
  "name": "invokeMyLambdaFunction",
  "action": "aws:invokeLambdaFunction",
  "maxAttempts": 3,
```

```
"timeoutSeconds": 120,
"onFailure": "Abort",
"inputs": {
  "FunctionName": "MyLambdaFunction"
}
}
```

FunctionName

Lambda 函数的名称。此函数必须存在。

类型：字符串

必需：是

限定词

函数版本或别名。

类型：字符串

必需：否

InvocationType

调用类型。默认为 RequestResponse。

类型：字符串

有效值：Event | RequestResponse | DryRun

必需：否

LogType

如果 Tail，调用类型必须为 RequestResponse。AWS Lambda 返回 Lambda 函数生成的采用 base64 编码的最后 4 KB 日志数据。

类型：字符串

有效值：None | Tail

必需：否

ClientContext

特定于客户端的信息。

必需：否

Payload

您的 Lambda 函数的 JSON 输入。

必需：否

输出

StatusCode

函数执行状态代码。

FunctionError

指示执行 Lambda 函数时是否出现错误。如果出现错误，则此字段将显示 Handled 或 Unhandled。Handled 错误由函数报告。Unhandled 错误由 AWS Lambda 检测和报告。

LogResult

Lambda 函数调用的 base64 编码日志。只有在调用类型为 `RequestResponse` 并且请求了日志时，才存在日志。

Payload

Lambda 函数返回的对象的 JSON 表示形式。只有在调用类型为 `RequestResponse` 时才存在有效负载。

aws:pause

此操作会暂停自动化执行。暂停后的执行状态为正在等待。要继续自动化执行，请使用信号类型为“恢复”的 [SendAutomationSignal](#) API 操作。

输入

输入如下。

```
{
  "name": "pauseThis",
  "action": "aws:pause",
  "inputs": {}
}
```

输出

无

aws:runCommand

运行指定的命令。

Note

Automation 仅支持一个 Run Command 操作的输出。一个文档可以包括多个 Run Command 操作和插件，但一次仅支持对一个操作和插件输出。

输入

此操作支持大多数 `send command` 参数。有关更多信息，请参阅 [SendCommand](#)。

```
{
  "name": "installPowerShellModule",
  "action": "aws:runCommand",
  "inputs": {
    "DocumentName": "AWS-InstallPowerShellModule",
    "InstanceIds": ["i-1234567890abcdef0"],
    "Parameters": {
      "source": "https://my-s3-url.com/MyModule.zip ",
      "sourceHash": "ASDFWER12321WRW"
    }
  }
}
```

DocumentName

Run Command 文档的名称。

类型：字符串

必需：是

InstanceIds

实例的 ID。

类型：字符串

必需：是

参数

文档中指定的必需参数和可选参数。

类型：映射

必需：否

评论

有关此命令的用户定义的信息。

类型：字符串

必需：否

DocumentHash

文档的哈希。

类型：字符串

必需：否

DocumentHashType

哈希的类型。

类型：字符串

有效值：Sha256 | Sha1

必需：否

NotificationConfig

用于发送通知的配置。

必需：否

OutputS3BucketName

命令执行响应的 S3 存储桶的名称。

类型：字符串

必需：否

OutputS3KeyPrefix

前缀。

类型：字符串

必需：否

ServiceRoleArn

IAM 角色的 ARN。

类型：字符串

必需：否

TimeoutSeconds

run-command 超时值 (以秒为单位)。

类型：整数

必需：否

输出

CommandId

命令的 ID。

Status

命令的状态。

ResponseCode

命令的响应代码。

输出

命令的输出。

aws:runInstances

启动新实例。

输入

此操作支持大多数 API 参数。有关更多信息，请参阅 [RunInstances](#) API 文档。

```
{
  "name": "launchInstance",
  "action": "aws:runInstances",
  "maxAttempts": 3,
  "timeoutSeconds": 1200,
  "onFailure": "Abort",
  "inputs": {
    "ImageId": "ami-12345678",
    "InstanceType": "t2.micro",
    "MinInstanceCount": 1,
    "MaxInstanceCount": 1,
    "IamInstanceProfileName": "myRunCmdRole"
  }
}
```

ImageId

Amazon 系统映像 (AMI) 的 ID。

类型：字符串

必需：是

InstanceType

实例的类型。

类型：字符串

必需：否

MinInstanceCount

要启动的实例的最小数量。

类型：字符串

必需：否

MaxInstanceCount

要启动的实例的最大数量。

类型：字符串

必需：否

AdditionalInfo

预留。

类型：字符串

必需：否

BlockDeviceMappings

适用于实例的块储存设备。

类型：MapList

必需：否

ClientToken

用于确保请求的幂等性的标识符。

类型：字符串

必需：否

DisableApiTermination

启用或禁用实例 API 终止

类型：布尔值

必需：否

EbsOptimized

启用或禁用 EBS 优化。

类型：布尔值

必需：否

IamInstanceProfileArn

实例的 IAM 实例配置文件的 ARN。

类型：字符串

必需：否

IamInstanceProfileName

实例的 IAM 实例配置文件的名称。

类型：字符串

必需：否

InstanceInitiatedShutdownBehavior

指示此实例是否在系统关闭时停止或终止。

类型：字符串

必需：否

KernelId

内核的 ID。

类型：字符串

必需：否

KeyName

密钥对的名称。

类型：字符串

必需：否

MaxInstanceCount

搜索服务时可筛选的实例数量上限。

类型：整数

必需：否

MinInstanceCount

搜索服务时可筛选的实例数量下限。

类型：整数

必需：否

监控

启动或禁用详细监控。

类型：布尔值

必需：否

NetworkInterfaces

网络接口。

类型：MapList

必需：否

Placement

实例的置放。

类型：StringMap

必需：否

PrivateIpAddress

主要 IPv4 地址。

类型：字符串

必需：否

RamdiskId

RAM 磁盘的 ID。

类型：字符串

必需：否

SecurityGroupIds

实例的安全组的 ID。

类型：StringList

必需：否

SecurityGroups

实例的安全组的名称。

类型：StringList

必需：否

SubnetId

子网 ID。

类型：字符串

必需：否

UserData

作为字符串文本值提供的执行脚本。如果输入文本值，则必须为 Base64 编码。

类型：字符串

必需：否

输出

InstanceIds

实例的 ID。

aws:sleep

将 Automation 执行时间延迟指定的时间。此操作使用国际标准化组织 (ISO) 8601 日期和时间格式。有关此日期和时间格式的更多信息，请参阅 [ISO 8601](#)。

输入

可将执行时间延迟指定的时间。

```
{
  "name": "sleep",
  "action": "aws:sleep",
  "inputs": {
    "Duration": "PT10M"
  }
}
```

还可以将执行时间延迟到指定日期和时间。如果指定日期和时间已过，操作将立即执行。

```
{
  "name": "sleep",
  "action": "aws:sleep",
  "inputs": {
    "Timestamp": "2020-01-01T01:00:00Z"
  }
}
```

Note

Automation 当前支持的最大延迟为 604800 秒 (7 天)。

Duration

ISO 8601 持续时间。您不能指定负数持续时间。

类型：字符串

必需：否

时间戳

ISO 8601 时间戳。如果您没有为此参数指定值，那么必须为 Duration 参数指定一个值。

类型：字符串

必需：否

输出

无

AWS Systems Manager Parameter Store

AWS Systems Manager Parameter Store 可提供安全的分层存储，用于配置数据管理和密钥管理。也可以将密码、数据库字符串和许可证代码等数据存储为参数值。可以将值存储为纯文本或加密数据。然后，可以使用创建参数时指定的唯一名称来引用对应值。Parameter Store 由 AWS 云支持，它可扩展、高度可用且持久。Parameter Store 不另外收取费用。

Parameter Store 提供以下优势和功能。

- 使用安全、可扩展的托管密钥管理服务 (无需管理服务器)。
- 通过将数据与代码分离来提高安全性。
- 分层存储配置数据和密钥字符串，而且可跟踪版本。
- 实现以细粒度控制和审核访问。
- 配置更改通知并触发自动操作。

- 单独标记参数，然后从不同级别进行安全访问，包括操作、参数、EC2 标签或路径级别。
- 跨 AWS 服务引用参数，如 Amazon EC2、Amazon Elastic Container Service、AWS Lambda、AWS CloudFormation、AWS CodeBuild、AWS CodeDeploy 和其他 Systems Manager 功能。
- 对于与 AWS KMS、Amazon SNS、Amazon CloudWatch 和 AWS CloudTrail 的集成进行配置，以实现加密、通知、监控和审核功能。

Systems Manager 参数入门

要开始使用 Systems Manager 参数，请完成以下任务。

任务	了解更多信息
了解如何使用不同类型的 Systems Manager 参数。	关于 Systems Manager 参数 (p. 362)
了解如何组织、创建和标记参数。	使用 Systems Manager 参数 (p. 365)
配置参数访问权限和通知。	设置 Systems Manager 参数 (p. 377)
了解如何在测试环境中创建和使用 Systems Manager 参数。	Systems Manager Parameter Store 演练 (p. 382)
了解 Parameter Store 如何使用 AWS KMS 管理 SecureString 参数。	AWS Systems Manager Parameter Store 如何使用 AWS KMS

相关内容

以下博客文章提供了关于 Parameter Store 以及如何借助其他 AWS 服务使用此功能的更多信息。

- 有关 Parameter Store 限制的信息，请参阅 Amazon Web Services General Reference 中的 [AWS Systems Manager 限制](#)。
- [使用 Parameter Store 和任务的 IAM 角色来管理 Amazon ECS 应用程序的密钥](#)
- [使用 Parameter Store 安全地访问 AWS CodeDeploy 中的密钥和配置数据](#)
- [有关 EC2 Systems Manager Parameter Store 的有趣文章](#)

关于 Systems Manager 参数

您可以在脚本、命令、配置和自动化流程中引用 Systems Manager 参数。参数与 Systems Manager 功能 (如 Run Command、State Manager 和 Automation) 结合使用。您还可以在其他 AWS 服务 (如 Amazon Elastic Container Service 和 AWS Lambda) 中引用参数。

使用 Systems Manager 功能，您可以在 AWS CLI 或 AWS Tools for Windows PowerShell 命令或脚本中引用 Systems Manager 参数。您还可以在 SSM 文档中引用参数。有关 SSM 文档的更多信息，请参阅 [AWS Systems Manager 文档 \(p. 289\)](#)。

主题

- [参数使用示例 \(p. 362\)](#)
- [使用安全字符串参数 \(p. 364\)](#)

参数使用示例

以下是一个在 Run Command 的 AWS CLI 命令中使用 Systems Manager 参数的示例。Systems Manager 参数始终带有前缀 `ssm:`。

```
aws ssm send-command --instance-ids i-1a2b3c4d5e6f7g8 --document-name AWS-RunPowerShellScript --parameter '{"commands":["echo {{ssm:parameter name}}"]}'
```

您还可以在 SSM 文档的参数部分引用 Systems Manager 参数，如下例所示。

```
{
  "schemaVersion": "2.0",
  "description": "Sample version 2.0 document v2",
  "parameters": {
    "commands": {
      "type": "StringList",
      "default": ["{{ssm:parameter name}}"]
    }
  },
  "mainSteps": [
    {
      "action": "aws:runShellScript",
      "name": "runShellScript",
      "inputs": {
        "runCommand": "{{commands}}"
      }
    }
  ]
}
```

Note

SSM 文档的 runtimeConfig 部分使用的是类似于本地参数的语法。本地参数不同于 Systems Manager 参数。您可以通过是否存在 ssm: 前缀区分本地参数和 Systems Manager 参数。

```
"runtimeConfig": {
  "aws:runShellScript": {
    "properties": [
      {
        "id": "0.aws:runShellScript",
        "runCommand": "{{ commands }}",
        "workingDirectory": "{{ workingDirectory }}",
        "timeoutSeconds": "{{ executionTimeout }}"
      }
    ]
  }
}
```

SSM 文档当前不支持对安全字符串参数的引用。这意味着，要通过 Run Command (举例来说) 使用安全字符串参数，您必须在将参数值传递到 Run Command 之前对它们进行检索，如以下示例所示：

AWS CLI

```
$value=aws ssm get-parameters --names the parameter name --with-decryption
```

```
aws ssm send-command -name AWS-JoinDomain -parameters password=$value -instance-id the instance ID
```

Tools for Windows PowerShell

```
$secure = (Get-SSMParameter -Names the parameter name -WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -AsPlainText -Force
```

```
$cred = New-Object System.Management.Automation.PSCredential -argumentlist user name, $secure
```

使用安全字符串参数

安全字符串参数是需要以安全的方式存储和引用的任何敏感数据。如果您有不希望用户更改或以明文形式引用的数据 (例如密码或许可证密钥), 则应使用安全字符串数据类型创建这些参数。我们建议您对以下情形使用安全字符串参数。

- 您想要在各种 AWS 服务间使用数据/参数, 但又不想以明文形式在命令、函数、代理日志或 AWS CloudTrail 日志中公开这些值。
- 您想要控制可以访问敏感数据的人。
- 您希望在有人访问敏感数据时进行审核 (AWS CloudTrail)。
- 您希望对您的敏感数据进行 AWS 级别的加密, 并想用自己的加密密钥管理访问。

如果您在创建参数时选择安全字符串数据类型, 则 AWS KMS 会加密参数值。有关更多信息, 请参阅 AWS Key Management Service Developer Guide 中的 [AWS Systems Manager Parameter Store 如何使用 AWS KMS](#)。

Important

只有安全字符串参数的值会被加密。参数的名称、描述和其他属性不会被加密。因此, 为了让包含密码的参数不那么明显, 我们建议使用在参数名称中避免“password”实际字样的命名系统。不过, 为了便于说明, 我们在示例中使用了 *my-password* 示例参数名称。

使用默认 KMS CMK 创建安全字符串参数

Systems Manager 包含默认 AWS KMS 客户主密钥 (CMK)。您可以通过 AWS CLI 执行以下命令, 以查看此密钥:

```
aws kms describe-key --key-id alias/aws/ssm
```

如果使用默认 KMS CMK 创建安全字符串参数, 则无需为 `--key-id` 参数提供值。以下 CLI 示例显示了在 Parameter Store 中不使用 `--key-id` 参数创建新的安全字符串参数的命令:

```
aws ssm put-parameter --name a_name --value "a value" --type SecureString
```

使用自定义 KMS CMK 创建安全字符串参数

如果需要使用自定义 KMS CMK 而非分配给您账户的默认 CMK, 则必须使用 `--key-id` 参数指定自定义 KMS CMK。该参数支持以下 AWS KMS 参数格式。

- 密钥 ARN 示例:

```
arn:aws:kms:us-east-2:123456789012:key/12345678-1234-1234-1234-123456789012
```

- 别名 ARN 示例:

```
arn:aws:kms:us-east-2:123456789012:alias/MyAliasName
```

- 全局唯一密钥 ID 示例:

```
12345678-1234-1234-1234-123456789012
```

- 别名示例:

```
alias/MyAliasName
```

您可以从 AWS CLI 使用以下命令创建自定义 AWS KMS CMK:

```
aws kms create-key
```

使用以下格式的命令，利用您刚刚创建的密钥创建安全字符串参数。

```
aws ssm put-parameter --name a_name --value "a value" --type SecureString --key-id arn:aws:kms:us-east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

Note

您可以使用加密值手动创建参数。在本例中，由于值已经加密，所以您无需选择安全字符串数据类型。如果您选择安全字符串，将对您的参数进行双重加密。

默认情况下，所有安全字符串值均显示为密码文本。要解密安全字符串值，用户必须有权调用 KMS [Decrypt](#) API 操作。有关配置 KMS 访问控制的信息，请参阅 [AWS Key Management Service Developer Guide](#) 中的 [AWS KMS 的身份验证和访问控制](#)。

将安全字符串参数用于其他 AWS 服务

您还可以在其他 AWS 服务中使用安全字符串参数。在以下示例中，Lambda 函数使用 [GetParameters](#) API 检索安全字符串参数。

```
from __future__ import print_function

import json
import boto3
ssm = boto3.client('ssm', 'us-east-2')
def get_parameters():
    response = ssm.get_parameters(
        Names=['LambdaSecureString'], WithDecryption=True
    )
    for parameter in response['Parameters']:
        return parameter['Value']

def lambda_handler(event, context):
    value = get_parameters()
    print("value1 = " + value)
    return value # Echo back the first key value
```

相关主题

有关如何创建和使用安全字符串参数的示例，请参阅[演练：创建安全字符串参数并将实例加入到域 \(PowerShell\)](#) (p. 385)。有关将 Systems Manager 参数与其他 AWS 服务结合使用的更多信息，请参阅以下博客文章。

- [使用 Parameter Store 和任务的 IAM 角色来管理 Amazon ECS 应用程序的密钥](#)
- [使用 Parameter Store 安全地访问 AWS CodeDeploy 中的密钥和配置数据](#)
- [有关 Amazon EC2 Systems Manager Parameter Store 的有趣文章](#)

使用 Systems Manager 参数

本部分介绍如何组织、创建和标记参数以及创建不同版本的参数。

主题

- [将参数组织成层次结构](#) (p. 366)
- [参数名称的要求和约束](#) (p. 368)

- [创建 Systems Manager 参数 \(p. 369\)](#)
- [标记 Systems Manager 参数 \(p. 373\)](#)
- [使用参数版本 \(p. 375\)](#)

将参数组织成层次结构

以平面列表的形式管理几十个或数百个参数十分耗时且容易出错。而且为任务确定正确参数也很难。这意味着，您可能意外地使用了错误的参数，或者您可能创建多个使用相同配置数据的参数。

您可以使用参数层次结构来帮助组织和管理参数。一个层次结构是一个参数名称，包括您使用正斜杠定义的路径。

Important

只有安全字符串参数的值会被加密。参数的名称、描述和其他属性不会被加密。因此，为了让包含密码的参数不那么明显，我们建议使用在参数名称中避免“password”实际字样的命名系统。不过，为了便于说明，我们在示例中使用了 *my-password* 示例参数名称。

下面是一个示例，它在名称中使用三个层次结构级别来标识以下内容：

/环境/计算机类型/应用程序/数据

```
/Dev/DBServer/MySQL/db-string13
```

您可以创建具有最多 15 个级别的层次结构。我们建议您创建反映环境中现有层次结构的层次结构，如以下示例所示：

- 您的[持续集成](#)和[持续交付](#)环境 (CI/CD 工作流)

```
/Dev/DBServer/MySQL/db-string
```

```
/Staging/DBServer/MySQL/db-string
```

```
/Prod/DBServer/MySQL/db-string
```

- 您的使用容器的应用程序

```
/MyApp/.NET/Libraries/my-password
```

- 您的业务组织

```
/Finance/Accountants/UserList
```

```
/Finance/Analysts/UserList
```

```
/HR/Employees/EU/UserList
```

参数层次结构规范了创建参数的方式，而且使得随时间的推移管理参数更为容易。参数层次结构还可帮助您为配置任务确定正确参数。这可帮助您避免使用相同的配置数据创建多个参数。

您可以创建一个层次结构，允许您在不同的环境中共享参数，如以下示例所示，这些示例在开发和暂存环境中使用密码。

```
/DevTest/MyApp/database/my-password
```

然后您可以为生产环境创建一个唯一密码，如以下示例所示：

```
/prod/MyApp/database/my-password
```

您无需指定参数层次结构。您可以在第一级创建参数。它们叫做根参数。考虑到向后兼容性，在发布层次结构之前在 Parameter Store 中创建的所有参数都是根参数。系统将以下两个参数视为根参数。

```
/parameter-name
```

```
parameter-name
```

有关如何使用参数层次结构的示例，请参阅[演练：使用层次结构管理参数 \(AWS CLI\)](#) (p. 386)。

查询层次结构中的参数

使用层次结构的另一个好处是，能够通过使用 [GetParametersByPath](#) API 操作查询层次结构中的所有参数。例如，如果您从 AWS CLI 执行以下命令，系统将返回 IIS 级别中的所有参数。

```
aws ssm get-parameters-by-path --path /Dev/Web/IIS
```

要查看层次结构中解密的 SecureString 参数，请指定路径和 `--with-decryption` 参数，如以下示例所示。

```
aws ssm get-parameters-by-path --path /Prod/ERP/SAP --with-decryption
```

使用层次结构限制 IAM 权限

通过对 Parameter Store API 操作使用层次结构和 AWS Identity and Access Management (IAM) 策略，您可以提供或限制对层次结构的一个级别中所有参数的访问权限。以下示例策略允许对 US East (Ohio) Region (us-east-2) 中 AWS 账户 123456789012 的所有参数执行所有 Parameter Store 操作。用户不能创建参数，因为 `PutParameter` 操作已被显式拒绝。此策略还禁止用户调用 `GetParametersByPath` 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:*"
      ],
      "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "ssm:GetParametersByPath"
      ],
      "Condition": {
        "StringEquals": {
          "ssm:Recursive": [
            "true"
          ]
        }
      },
      "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/Dev/ERP/Oracle/*"
    }
  ]
}
```



```
        "Effect": "Deny",
        "Action": [
            "ssm:PutParameter"
        ],
        "Condition": {
            "StringEquals": {
                "ssm:Overwrite": [
                    "false"
                ]
            }
        },
        "Resource": "arn:aws:ssm:us-east-2:123456789012:parameter/*"
    }
}
```

Important

如果用户有权访问某个路径，则该用户可以访问该路径的所有级别。例如，如果用户有权访问路径 /a，则该用户也可以访问 /a/b。即使用户在 IAM 中被明确拒绝对参数 /a 的访问，他们仍然可以递归调用 GetParametersByPath API 操作并查看 /a/b。

参数名称的要求和约束

使用本主题中的信息可在您创建参数时帮助您为参数名称指定有效值。

此信息对 AWS Systems Manager API Reference 中的主题 [PutParameter](#) 中的详细信息进行了补充，该主题提供了有关 AllowedPattern、Description、KeyId、Overwrite、Type 和 Value 值的信息。

参数名称的要求和约束包括：

- 区分大小写：参数名称区分大小写。
- 空格：参数名称不能包含空格。
- 有效字符：参数名称只能包含以下符号和字母：a-zA-Z0-9_.-/
- 前缀：参数名称不得以“aws”或“ssm”（不区分大小写）作为前缀。例如，尝试使用以下名称创建参数将失败，并引发异常：
 - awsTestParameter
 - SSM-testparameter
 - /aws/testparam1

Note

在 SSM 文档、命令或脚本中指定参数时，可将 ssm 作为语法的一部分包含，如以下示例中所示。请注意，括号之间没有空格。

- 有效：{{ssm:parameter_name}} 和 {{ssm:parameter_name}}，例如 {{ssm:addUsers}} 和 {{ssm:addUsers }}，
- 无效：{{ssm:ssmAddUsers}}
- 唯一性：参数名称必须在 AWS 区域中是唯一的。例如，Systems Manager 将以下参数视为不同的参数（如果它们存在于同一区域中）：
 - /Test/TestParam1
 - /TestParam1

以下示例也是唯一的：

- /Test/TestParam1/Logpath1
- /Test/TestParam1

不过，以下示例（如果在同一区域中）不是唯一的：

- /TestParam1
 - TestParam1
 - 层次结构深度：如果指定参数层次结构，则层次结构可以具有最多十五个级别的深度。可以在该层次结构的任何级别定义参数。以下两个示例的结构都是有效的：
 - /Level-1/L2/L3/L4/L5/L6/L7/L8/L9/L10/L11/L12/L13/L14/parameter-name
 - parameter-name
- 尝试创建以下参数将失败并引发 `HierarchyLevelLimitExceededException` 异常：
- /Level-1/L2/L3/L4/L5/L6/L7/L8/L9/L10/L11/L12/L13/L14/L15/L16/parameter-name

Important

如果用户有权访问某个路径，则该用户可以访问该路径的所有级别。例如，如果用户有权访问路径 /a，则该用户也可以访问 /a/b。即使用户在 IAM 中被明确拒绝对参数 /a 的访问，他们仍然可以递归调用 [GetParametersByPath](#) API 操作并查看 /a/b。

创建 Systems Manager 参数

使用以下主题中的信息可帮助您使用 AWS CLI、AWS Tools for Windows PowerShell、或 AWS Systems Manager 控制台创建 Systems Manager 参数。

主题

- [创建 Systems Manager 参数 \(AWS CLI\)](#) (p. 369)
- [创建 Systems Manager 参数 \(Tools for Windows PowerShell\)](#) (p. 371)
- [创建 Systems Manager 参数 \(控制台\)](#) (p. 372)

创建 Systems Manager 参数 (AWS CLI)

可以使用 AWS CLI 创建一个使用 `String`、`StringList` 或 `SecureString` 数据类型的参数。

有关使用 AWS CLI 创建参数的更多信息，请参阅[演练：在命令中创建和使用参数 \(AWS CLI\)](#) (p. 382)。

Note

参数只在创建它的区域可用。

主题

- [创建 String 或 StringList 参数 \(AWS CLI\)](#) (p. 369)
- [创建一个 SecureString 参数 \(AWS CLI\)](#) (p. 370)

创建 String 或 StringList 参数 (AWS CLI)

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
```

```
Default output format [None]: ENTER
```

2. 执行以下命令创建一个参数。

```
aws ssm put-parameter --name "a_name" --value "a value, or a comma-separated list of values" --type String or StringList
```

如果成功，则该命令返回参数的版本号。

下面是使用 StringList 数据类型的示例。

```
aws ssm put-parameter --name /IAD/ERP/Oracle/addUsers --value "Milana,Mariana,Mark,Miguel" --type StringList
```

Note

StringList 中的项目必须用逗号 (,) 分隔。不能使用其他标点符号或特殊字符对列表中的项目进行转义。如果您有需要逗号的参数值，则使用 String 数据类型。

3. 执行以下命令验证参数的详细信息。

```
aws ssm get-parameters --name "the name you specified"
```

下面是使用前面示例中指定的名称的示例。

```
aws ssm get-parameters --name "/IAD/ERP/Oracle/addUsers"
```

创建一个 SecureString 参数 (AWS CLI)

在创建 SecureString 参数前，请阅读关于此类型参数的要求。有关更多信息，请参阅 [使用安全字符串参数 \(p. 364\)](#)。

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon Elastic Compute Cloud (Amazon EC2) 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令创建一个参数。

```
aws ssm put-parameter --name "a_name" --value "a value" --type SecureString --key-id "a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN"
```

Note

要使用分配给您的账户的默认 AWS KMS CMK，请从命令中删除 key-id 参数。

下面是对密码参数和自定义 AWS KMS 密钥使用模糊名称 (elixir3131) 的示例。

```
aws ssm put-parameter --name /Finance/Payroll/elixir3131 --  
value "P@sSwJrd" --type SecureString --key-id arn:aws:kms:us-  
east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

3. 执行以下命令验证参数的详细信息。

```
aws ssm get-parameters --name "the name you specified" --with-decryption
```

Note

如果您不指定 `with-decryption` 参数，或者如果您指定 `no-with-decryption` 参数，命令会返回加密的 GUID。

创建 Systems Manager 参数 (Tools for Windows PowerShell)

可以使用 Tools for Windows PowerShell 创建一个使用 `String`、`StringList` 或 `SecureString` 数据类型的参数。

Note

参数只在创建它的区域可用。

主题

- [创建 String 或 StringList 参数 \(Tools for Windows PowerShell\) \(p. 371\)](#)
- [创建 SecureString 参数 \(Tools for Windows PowerShell\) \(p. 372\)](#)

创建 String 或 StringList 参数 (Tools for Windows PowerShell)

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 `us-east-2` 区域。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 执行以下命令创建一个参数。

```
Write-SSMParameter -Name "a name" -Value "a value, or a comma-separated list of values"  
-Type "String or StringList"
```

如果成功，则该命令返回参数的版本号。

Note

`StringList` 中的项目必须用逗号 (,) 分隔。不能使用其他标点符号或特殊字符对列表中的项目进行转义。如果您有需要逗号的参数值，则使用 `String` 数据类型。

下面是使用 `String` 数据类型的示例。

```
Write-SSMParameter -Name "/IAD/Web/SQL/IPaddress" -Value "99.99.99.999" -Type "String"
```

4. 执行以下命令验证参数的详细信息。

```
(Get-SSMParameterValue -Name "the name you specified").Parameters
```

创建 SecureString 参数 (Tools for Windows PowerShell)

在创建 SecureString 参数前，请阅读关于此类型参数的要求。有关更多信息，请参阅 [使用安全字符串参数 \(p. 364\)](#)。

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 us-east-2 区域。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 执行以下命令创建一个参数。

```
Write-SSMParameter -Name "a name" -Value "a value" -Type "SecureString" -KeyId "a KMS  
CMK ID, a KMS CMK ARN, an alias name, or an alias ARN"
```

如果成功，则该命令返回参数的版本号。

Note

要使用分配给您的账户的默认 AWS KMS CMK，请从命令中删除 -KeyId 参数。

下面是对密码参数和用户默认 KMS CMK 使用模糊名称 (elixir3131) 的示例。

```
Write-SSMParameter -Name "/Finance/Payroll/elixir3131" -Value "P@ssW)rd" -Type  
"SecureString"
```

4. 执行以下命令验证参数的详细信息。

```
(Get-SSMParameterValue -Name "the name you specified" -WithDecryption $true).Parameters
```

创建 Systems Manager 参数 (控制台)

可以使用 AWS Systems Manager 控制台创建 Systems Manager 参数。

Note

参数只在创建它的区域可用。

创建参数 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Parameter Store。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Parameter Store。

3. 选择创建参数。
4. 对于 Name，键入层次结构和参数名称。例如，键入 /Test/helloWorld。
5. 在 Description 框中，键入用于将此参数标识为测试参数的描述。
6. 对于 Type，选择 String、StringList 或 SecureString。

Note

如果选择 SecureString，则会显示 KMS Key ID 字段。如果不提供 KMS CMK ID、KMS CMK ARN、别名或别名 ARN，则系统使用 alias/aws/ssm (默认)，这是 Systems Manager 的默认 KMS CMK。如果您不想使用此密钥，则可以选择自定义密钥。有关更多信息，请参阅 [使用安全字符串参数 \(p. 364\)](#)。

7. 在 Value 框中，键入一个值。例如，键入 MyFirstParameter。如果您选择 Secure String，则在您键入时值会被遮蔽。
8. 选择创建参数。
9. 在参数列表中，选择刚才创建的参数的名称。验证 Overview 选项卡上的详细信息。如果您创建了 SecureString 参数，请选择 Show 查看未加密值。

标记 Systems Manager 参数

可以使用 Systems Manager 控制台、AWS CLI、AWS Tools for Windows 或 [AddTagsToResource](#) API 向 Systems Manager 资源 (包括文档、托管实例、Maintenance Window、Parameter Store 参数和补丁基准) 添加标签。

标签用于对参数进行组织。例如，可以为特定环境、部门或用户和组标记参数。在标记参数之后，可以通过创建一个指定用户可访问的标签的 IAM 策略来限制对该参数的访问权限。有关使用标签限制对参数的访问权限的更多信息，请参阅 [使用标签控制对参数的访问权限 \(p. 380\)](#)。

有关提供 Systems Manager 的区域的信息，请参阅 [区域](#)。

主题

- [标记参数 \(控制台\) \(p. 373\)](#)
- [标记参数 \(AWS CLI\) \(p. 373\)](#)
- [标记参数 \(AWS Tools for Windows\) \(p. 374\)](#)

标记参数 (控制台)

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在左侧导航窗格中，选择 Parameter Store。
3. 选择您创建的参数的名称，然后选择 Tags 选项卡。
4. 在第一个框中，为标签输入键，例如 **Environment**。
5. 在第二个框中，为标签输入值，例如 **Test**。
6. 选择 Save。

标记参数 (AWS CLI)

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令列出您可以标记的参数。

```
aws ssm describe-parameters
```

记下您想标记的参数的名称。

3. 执行以下命令标记一个参数。

```
aws ssm add-tags-to-resource --resource-type "Parameter" --resource-id "the parameter name" --tags "Key=a key, for example Environment,Value=a value, for example TEST"
```

如果成功，则命令没有输出。

4. 执行以下命令验证参数标签。

```
aws ssm list-tags-for-resource --resource-type "Parameter" --resource-id "the parameter name"
```

标记参数 (AWS Tools for Windows)

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件](#) (p. 6)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 us-east-2 区域。Systems Manager 目前在以下 [区域](#) 可用。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 执行以下命令列出您可以标记的参数。

```
Get-SSMParameterList
```

4. 执行以下命令标记一个参数。

```
$tag1 = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag1.Key = "Environment"
$tag1.Value = "TEST"
Add-SSMResourceTag -ResourceType "Parameter" -ResourceId "the parameter name" -Tag $tag1
```

如果成功，则命令没有输出。

5. 执行以下命令验证参数标签。

```
Get-SSMResourceTag -ResourceType "Parameter" -ResourceId "the parameter name"
```

使用参数版本

在最初创建一个参数时，Parameter Store 为该参数分配版本 1。每次编辑参数时，Parameter Store 自动将其版本号增加 1。您可以在 API 调用和 SSM 文档中指定参数名和特定版本号。如果不指定版本号，系统自动使用最新版本。

参数版本针对意外更改参数的情况提供额外保护。您可以查看详细信息，包括所有版本的值。还可以使用参数版本查看一段时间内更改参数的次数。

可以使用以下格式，在命令、API 调用和 SSM 文档中引用包括旧版本在内的特定参数版本：
`{{ssm:parameter_name:version}}`。以下示例是关于在 SSM 文档中指定的、名为 RunCommand 的参数：

```
{
  "schemaVersion": "1.2",
  "description": "Run a shell script or specify the commands to run.",
  "parameters": {
    "commands": {
      "type": "StringList",
      "description": "(Required) Specify a shell script or a command to run.",
      "minItems": 1,
      "displayType": "textarea",
      "default": "{{ssm:RunCommand:2}}"
    },
    "executionTimeout": {
      "type": "String",
      "default": "3600",
      "description": "(Optional) The time in seconds for a command to complete before it is considered to have failed. Default is 3600 (1 hour). Maximum is 172800 (48 hours).",
      "allowedPattern": "([1-9][0-9]{0,3})|(1[0-9]{1,4})|(2[0-7][0-9]{1,3})|(28[0-7][0-9]{1,2})|(28800)"
    }
  },
  "runtimeConfig": {
    "aws:runShellScript": {
      "properties": [
        {
          "id": "0.aws:runShellScript",
          "runCommand": "{{ commands }}",
          "timeoutSeconds": "{{ executionTimeout }}"
        }
      ]
    }
  }
}
```

以下过程向您演示如何编辑参数，并在之后验证是否已创建新版本。

主题

- [创建新的参数版本 \(控制台\) \(p. 375\)](#)
- [创建新的参数版本 \(AWS CLI\) \(p. 376\)](#)
- [创建新的参数版本 \(Windows PowerShell\) \(p. 377\)](#)

创建新的参数版本 (控制台)

可以使用 [AWS Systems Manager 控制台](#) 创建一个新版本的参数。

创建新的参数版本 ()

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。

2. 在导航窗格中，选择 Parameter Store。

-或者-

如果 AWS Systems Manager 主页首先打开，请选择菜单图标 (≡) 以打开导航窗格，然后选择 Parameter Store。

3. 选择之前创建的参数的名称。有关创建新参数的信息，请参阅[创建 Systems Manager 参数 \(p. 369\)](#)。

Note

参数只在创建它的区域可用。如果看不到要更新的参数，请验证您的区域。

4. 选择 Edit。
5. 在 Value 框中键入新的值，然后选择 Save changes。
6. 在参数列表中选择刚才更新的参数的名称，然后查看 History 选项卡。在 Overview 选项卡上，验证版本号是否增加 1，并验证新值。

创建新的参数版本 (AWS CLI)

执行以下过程，使用 AWS CLI 创建参数的新版本。

1. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅[Systems Manager 先决条件 \(p. 6\)](#)。

```
aws configure
```

系统将提示您指定以下内容：

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

2. 执行以下命令列出您可以更新的参数。

Note

参数只在创建它的区域可用。如果看不到要更新的参数，请验证您的区域。

```
aws ssm describe-parameters
```

记下您想更新的参数的名称。

3. 执行以下命令更新参数值。

```
aws ssm put-parameter --name "the_parameter_name" --type the_parameter_type --value
"the_new_value" --overwrite
```

4. 执行以下命令，查看该参数的所有版本。

```
aws ssm get-parameter-history --name "the_parameter_name"
```

5. 执行以下命令，按版本号检索有关参数的信息。

```
aws ssm get-parameters --names "the_parameter_name:the_version_number"
```

以下是示例。

```
aws ssm get-parameters --names "/Production/SQLConnectionString:3"
```

创建新的参数版本 (Windows PowerShell)

执行以下过程，使用 AWS Tools for Windows PowerShell 创建参数的新版本。

1. 打开 AWS Tools for Windows PowerShell 并执行以下命令指定您的凭证。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 IAM 中被授予相应权限。有关更多信息，请参阅 [Systems Manager 先决条件](#) (p. 6)。

```
Set-AWSCredentials -AccessKey key_name -SecretKey key_name
```

2. 执行以下命令为 PowerShell 会话设置区域。该示例使用 us-east-2 区域。Systems Manager 目前在以下 [区域](#) 可用。

```
Set-DefaultAWSRegion -Region us-east-2
```

3. 执行以下命令列出您可以更新的参数。

Note

参数只在创建它的区域可用。如果看不到要更新的参数，请验证您的区域。

```
Get-SSMParameterList
```

记下您想更新的参数的名称。

4. 执行以下命令更新参数值。

```
Write-SSMParameter -Name "the_parameter_name" -Value "the_new_value" -Type "the_parameter_type" -Overwrite $true
```

5. 执行以下命令，查看该参数的所有版本。

```
Get-SSMParameterHistory -Name "the_parameter_name"
```

6. 执行以下命令，按版本号检索有关参数的信息。

```
(Get-SSMParameterValue -Names "the_parameter_name").Parameters | fl
```

设置 Systems Manager 参数

要设置 Systems Manager 参数，您需要配置 AWS Identity and Access Management (IAM) 角色，这样 Systems Manager 才有权执行您为服务指定的操作。此部分提供有关如何使用 IAM 控制台手动配置这些角色的信息。此部分还介绍了如何创建 Amazon CloudWatch Events 事件以接收有关 Systems Manager 参数操作的通知。

内容

- [控制对 Systems Manager 参数的访问权限](#) (p. 378)
- [为 Systems Manager 参数设置通知和事件](#) (p. 381)

控制对 Systems Manager 参数的访问权限

您将使用 AWS Identity and Access Management (IAM) 控制对 Systems Manager 参数的访问权限。更具体地说，您将创建 IAM 策略来限制对以下 API 操作的访问权限：

- [DeleteParameter](#)
- [DeleteParameters](#)
- [DescribeParameters](#)
- [GetParameter](#)
- [GetParameters](#)
- [GetParameterHistory](#)
- [GetParametersByPath](#)
- [PutParameter](#)

我们建议您通过创建限制性 IAM 策略来控制对 Systems Manager 参数的访问权限。例如，以下策略允许您为特定资源调用 `DescribeParameters` 和 `GetParameters` API 操作。这意味着，您可以获取有关以 `prod-*` 开头的所有参数的信息并使用这些参数。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeParameters"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": "arn:aws:ssm:us-east-2:123456123:parameter/prod-*"
    }
  ]
}
```

对于受信任的管理员，您可以通过使用类似以下示例的策略提供对所有 Systems Manager 参数 API 操作的完全访问权限。此策略将授予用户对所有以 `dbserver-prod-*` 开头的生产参数的完全访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter",
        "ssm:DeleteParameter",
        "ssm:GetParameterHistory",
        "ssm:GetParametersByPath",
        "ssm:GetParameters",
        "ssm:GetParameter",
        "ssm:DeleteParameters"
      ],
      "Resource": "arn:aws:ssm:region:account_ID:parameter/dbserver-prod-*"
    }
  ]
}
```

```
    },  
    {  
      "Sid": "VisualEditor1",  
      "Effect": "Allow",  
      "Action": "ssm:DescribeParameters",  
      "Resource": "*"   
    }  
  ]  
}
```

主题

- [仅允许特定参数在实例上运行 \(p. 379\)](#)
- [使用标签控制对参数的访问权限 \(p. 380\)](#)

仅允许特定参数在实例上运行

您还可以控制访问权限，以使实例只能运行特定参数。以下示例支持实例获取仅用于以“prod-”开头的参数的参数值。如果参数是安全字符串，则实例会使用 AWS KMS 解密该字符串。

Note

如果您在创建参数时选择安全字符串数据类型，则 AWS KMS 会加密参数值。有关 AWS KMS 的更多信息，请参阅 [AWS Key Management Service Developer Guide](#)。

每个 AWS 账户都分配有默认 AWS KMS 密钥。您可以通过 AWS CLI 执行以下命令，以查看您的密钥：

```
aws kms describe-key --key-id alias/aws/ssm
```

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "ssm:GetParameters"  
      ],  
      "Resource": [  
        "arn:aws:ssm:region:account-id:parameter/prod-*"  
      ]  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "kms:Decrypt"  
      ],  
      "Resource": [  
        "arn:aws:kms:region:account-id:key/CMK"  
      ]  
    }  
  ]  
}
```

Note

如上述实例所示，实例策略会分配到 IAM 中的实例角色中。要详细了解如何配置对 Systems Manager 功能的访问权限，包括如何将策略分配给用户和实例，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

使用标签控制对参数的访问权限

在标记参数之后，可以通过创建一个指定用户可访问的标签的 IAM 策略来限制对该参数的访问权限。当用户尝试使用参数时，系统会检查 IAM 策略和为该参数指定的标签。如果用户对分配给该参数的标签没有访问权限，用户会收到访问被拒绝错误。

目前，您可以限制对以下 GetParameter API 操作的访问：

- [GetParameter](#)
- [GetParameters](#)
- [GetParameterHistory](#)

使用以下过程创建一个 IAM 策略来通过使用标签限制对参数的访问权限。

开始前的准备工作

创建并标记参数。有关更多信息，请参阅 [设置 Systems Manager 参数 \(p. 377\)](#)。

使用标签限制用户对参数的访问权限

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. 在导航窗格中选择 Policies，然后选择 Create policy。
3. 选择 JSON 选项卡。
4. 将以下示例策略复制粘贴到此文本字段中，替换示例文本。将 `tag_key` 和 `tag_value` 替换为您的标签的键值对。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetParameters"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/tag_key": [
            "tag_value"
          ]
        }
      }
    }
  ]
}
```

此示例策略仅限制对 GetParameters API 操作的访问。您可以在 Action 块中使用以下格式来限制对多个 API 操作的访问：

```
"Action": [
  "ssm:GetParameters",
  "ssm:GetParameter",
  "ssm:GetParameterHistory",
],
```

您可以使用以下 Condition 格式在策略中指定多个键。指定多个键会为这些键创建 AND 关系。

```
"Condition": {
```

```
"StringLike":{
  "ssm:resourceTag/tag_key1":[
    "tag_value1"
  ],
  "ssm:resourceTag/tag_key2":[
    "tag_value2"
  ]
}
```

您可以使用以下 Condition 格式在策略中指定多个值。ForAnyValue 为这些值建立 OR 关系。还可以指定ForAllValues 来建立 AND 关系。

```
"Condition":{
  "ForAnyValue:StringLike":{
    "ssm:resourceTag/tag_key1":[
      "tag_value1",
      "tag_value2"
    ]
  }
}
```

5. 选择查看策略。
6. 在 Name 字段中，指定一个将它标识为目标参数的用户策略的名称。
7. 输入描述。
8. 在摘要部分，验证策略详细信息。
9. 选择 Create policy。
10. 将策略分配给 IAM 用户或组。有关更多信息，请参阅[更改 IAM 用户的权限](#)和[将策略附加到 IAM 组](#)。

将策略附加到 IAM 用户或组账户之后，如果用户尝试使用参数，并且用户的策略不允许用户访问该参数的标签 (调用 GetParameters API)，系统会返回错误。错误类似于以下内容：

User: `user_name` isn't authorized to perform: `ssm:GetParameters` on resource: `parameter ARN` with the following command.

当参数具有多个标签时，如果用户无权访问这些标签中的任何一个，则用户仍会收到访问被拒绝错误。

为 Systems Manager 参数设置通知和事件

您可以使用 Amazon CloudWatch Events 和 Amazon SNS 来告知您有关 Systems Manager 参数的更改。在创建、更新或删除参数时，您会收到通知。

您还可以使用 CloudWatch 对特定参数事件的目标执行操作。这意味着，例如，您可以执行 AWS Lambda 函数在参数被删除后重新创建它。您还可以设置一个通知，以在数据库密码被更新时触发 Lambda 函数。Lambda 函数可以强制您的数据库连接重置或使用新密码重新连接。

开始前的准备工作

创建 Amazon SNS 主题。有关更多信息，请参阅Amazon Simple Notification Service Developer Guide中的[Amazon SNS 入门](#)。

为 Systems Manager 参数配置 CloudWatch Events

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. 在左侧导航窗格中，选择 Events，然后选择 Create rule。
3. 在 Event Source 下，验证已选中 Event Pattern。
4. 在 Service Name 字段中，选择 EC2 Simple Systems Manager (SSM)

5. 在 Event Type 字段中，选择 Parameter Store。
6. 选择您要接收通知的详细类型和状态，然后选择 Add targets。
7. 在 Targets 列表中，选择一个目标类型。例如，选择 Lambda 函数或选择 SNS 主题。有关不同目标类型的信息，请参阅对应的 AWS 帮助文档。
8. 在页面上向下滚动，然后选择 Configure details。
9. 指定规则详细信息，然后选择 Create rule。

Systems Manager Parameter Store 演练

以下演练向您展示如何使用 Parameter Store 在测试环境中创建、存储和执行参数。以下演练说明如何将 Parameter Store 与其他 Systems Manager 功能结合使用。此外，您还可以将 Parameter Store 与其他 AWS 服务结合使用。有关更多信息，请参阅 [将安全字符串参数用于其他 AWS 服务 \(p. 365\)](#)。

内容

- [演练：在命令中创建和使用参数 \(控制台\) \(p. 382\)](#)
- [演练：在命令中创建和使用参数 \(AWS CLI\) \(p. 382\)](#)
- [演练：创建安全字符串参数并将实例加入到域 \(PowerShell\) \(p. 385\)](#)
- [演练：使用层次结构管理参数 \(AWS CLI\) \(p. 386\)](#)

演练：在命令中创建和使用参数 (控制台)

以下过程将指导您完成在 Parameter Store 中创建参数然后执行使用此参数的命令的过程。

使用 Parameter Store 创建参数

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Parameter Store。
3. 选择创建参数。
4. 在 Name 框中，键入层次结构和名称。例如，键入 /Test/helloWorld。

有关参数层次结构的更多信息，请参阅[将参数组织成层次结构 \(p. 366\)](#)。

5. 在描述字段中，键入用于将此参数标识为测试参数的描述。
6. 对于类型，选择字符串。
7. 在 Value 字段中，键入一个字符串。例如，键入 My1stParameter。
8. 选择创建参数。
9. 在导航窗格中，选择 Run Command。
10. 选择运行命令。
11. 在 命令文档列表中，选择 AWS-RunPowershellScript (Windows) 或 AWS-RunShellScript (Linux)。
12. 在 Target instances 下，选择之前创建的实例。
13. 在 Commands 字段中，键入 echo {{ssm:*parameter name*}}，例如 echo {{ssm:/Test/helloWorld}}。
14. 选择 Run。
15. 滚动到 Command ID 页面底部，选择实例 ID 旁边的单选按钮，然后选择 View output。

演练：在命令中创建和使用参数 (AWS CLI)

以下过程将指导您完成使用 AWS CLI 创建和存储参数的过程。

使用 Parameter Store 创建字符串参数

1. 将 AWS CLI [下载](#)到本地计算机上。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令，使用字符串数据类型创建参数。--name 参数使用层次结构。有关层次结构的更多信息，请参阅[将参数组织成层次结构 \(p. 366\)](#)。

```
aws ssm put-parameter --name "a_name" --value "a value" --type String
```

下面是在名称中使用参数层次结构的示例。有关参数层次结构的更多信息，请参阅[将参数组织成层次结构 \(p. 366\)](#)。

```
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "My1stParameter" --type String
```

该命令返回参数的版本号。

4. 执行以下命令，查看参数元数据。

```
aws ssm describe-parameters --filters "Key=Name,Values=/Test/IAD/helloWorld"
```

Note

名称必须大写。

系统将返回类似于以下内容的信息。

```
{
  "Parameters": [
    {
      "LastModifiedUser": "arn:aws:iam::123456789:user/User's name",
      "LastModifiedDate": 1494529763.156,
      "Type": "String",
      "Name": "helloworld"
    }
  ]
}
```

5. 执行以下命令，更改参数值。

```
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "good day sunshine" --type String --overwrite
```

该命令返回参数的版本号。

6. 执行以下命令，查看最新的参数值。


```
aws ssm get-parameters --names "/Test/IAD/helloWorld"
```

系统将返回类似于以下内容的信息。

```
{
  "InvalidParameters": [],
  "Parameters": [
    {
      "Type": "String",
      "Name": "/Test/IAD/helloWorld",
      "Value": "good day sunshine"
    }
  ]
}
```

7. 执行以下命令，查看参数值的历史记录。

```
aws ssm get-parameter-history --name "/Test/IAD/helloWorld"
```

8. 执行以下命令，在命令中使用该参数。

```
aws ssm send-command --document-name "AWS-RunShellScript" --parameters '{"commands":
["echo {{ssm:/Test/IAD/helloWorld}}"]}' --targets "Key=instanceids,Values=instance_IDs"
```

使用以下过程创建安全字符串参数。有关安全字符串参数的更多信息，请参阅[使用安全字符串参数 \(p. 364\)](#)。

使用 AWS CLI 创建安全字符串参数

1. 执行下列命令之一以使用安全字符串数据类型创建参数。

创建使用默认 KMS 密钥的安全字符串参数

```
aws ssm put-parameter --name "a_name" --value "a value, for example P@ssW%rd#1" --type
"SecureString"
```

创建使用自定义 AWS KMS 密钥的安全字符串参数

```
aws ssm put-parameter --name "a_name" --value "a value" --type "SecureString" --key-id
"your AWS user account ID/the custom AWS KMS key"
```

下面是使用自定义 AWS KMS 密钥的示例。

```
aws ssm put-parameter --name "my-password" --value "P@ssW%rd#1" --type "SecureString"
--key-id "arn:aws:kms:us-east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e"
```

Important

只有安全字符串参数的值会被加密。参数的名称、描述和其他属性不会被加密。因此，为了让包含密码的参数不那么明显，我们建议使用在参数名称中避免“password”实际字样的命名系统。不过，为了便于说明，我们在示例中使用了 *my-password* 示例参数名称。

2. 执行以下命令，查看参数元数据。

```
aws ssm describe-parameters --filters "Key=Name,Values=the name that you specified"
```

3. 执行以下命令，更改参数值。

```
aws ssm put-parameter --name "the name that you specified" --value "new value" --type "SecureString" --overwrite
```

更新使用默认 KMS 密钥的安全字符串参数

```
aws ssm put-parameter --name "the name that you specified" --value "new value" --type "SecureString" --key-id "the AWS KMS key ID" --overwrite
```

更新使用自定义 KMS 密钥的安全字符串参数

```
aws ssm put-parameter --name "the name that you specified" --value "new value" --type "SecureString" --key-id "your AWS user account alias/the custom KMS key" --overwrite
```

4. 执行以下命令，查看最新的参数值。

```
aws ssm get-parameters --names "the name that you specified" --with-decryption
```

5. 执行以下命令，查看参数值的历史记录。

```
aws ssm get-parameter-history --name "the name that you specified"
```

Important

只有安全字符串参数的值会被加密。参数的名称、描述和其他属性不会被加密。因此，为了让包含密码的参数不那么明显，我们建议使用在参数名称中避免“password”实际字样的命名系统。不过，为了便于说明，我们在示例中使用了 *my-password* 示例参数名称。

演练：创建安全字符串参数并将实例加入到域 (PowerShell)

本演练介绍如何使用 Systems Manager 安全字符串参数和 Run Command 将 Windows 实例加入到域中。演练中使用了典型的域参数，例如 DNS 地址、域名和域用户名。这些值作为未加密的字符串值传递。将使用 AWS KMS 主密钥加密域密码并将其作为安全字符串传递。

创建安全字符串参数并将实例加入域

1. 使用 AWS Tools for Windows PowerShell 将参数输入到系统中。

```
Write-SSMParameter -Name DNS-IP -Value a DNS IP address -Type String  
Write-SSMParameter -Name domainName -Value the domain name -Type String  
Write-SSMParameter -Name domainJoinUserName -Value a user name -Type String  
Write-SSMParameter -Name my-password -Value a password -Type SecureString
```

Important

只有安全字符串参数的值会被加密。参数的名称、描述和其他属性不会被加密。因此，为了让包含密码的参数不那么明显，我们建议使用在参数名称中避免“password”实际字样的命名系统。不过，为了便于说明，我们在示例中使用了 *my-password* 示例参数名称。

2. 将 AmazonEC2RoleforSSM 托管策略附加到您的实例的 IAM 角色权限。有关信息，请参阅[托管策略与内联策略](#)。
3. 编辑附加到实例的 IAM 角色并添加以下策略。此策略授予实例调用 kms:Decrypt API 的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:region:account_id:key/key_id"
      ]
    }
  ]
}
```

4. 复制以下 json 示例并粘贴到简单文本编辑器中，并将文件保存为以下位置中的 JoinInstanceToDomain.json：c:\temp\JoinInstanceToDomain.json。

```
{
  "schemaVersion": "2.0",
  "description": "Run a PowerShell script to securely domain-join a Windows instance",
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "runPowerShellWithSecureString",
      "inputs": {
        "runCommand": [
          "$ipdns = (Get-SSMParameterValue -Name dns).Parameters[0].Value\n",
          "$domain = (Get-SSMParameterValue -Name domainName).Parameters[0].Value\n",
          "$username = (Get-SSMParameterValue -Name domainJoinUserName).Parameters[0].Value\n",
          "$password = (Get-SSMParameterValue -Name domainJoinPassword -WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -asPlainText -Force\n",
          "$credential = New-Object System.Management.Automation.PSCredential($username,$password)\n",
          "Set-DnsClientServerAddress \"Ethernet 2\" -ServerAddresses $ipdns\n",
          "Add-Computer -DomainName $domain -Credential $credential\n",
          "Restart-Computer -force"
        ]
      }
    }
  ]
}
```

5. 在 AWS Tools for Windows PowerShell 中执行以下命令以创建新 SSM 文档。

```
$json = Get-Content C:\temp\JoinInstanceToDomain | Out-String
New-SSMDocument -Name JoinInstanceToDomain -Content $json -DocumentType Command
```

6. 在 AWS Tools for Windows PowerShell 中执行以下命令以将实例加入到域

```
Send-SSMCommand -InstanceId Instance-ID -DocumentName JoinInstanceToDomain
```

演练：使用层次结构管理参数 (AWS CLI)

此演练向您说明如何通过使用 AWS CLI 来使用参数和参数层次结构参数。有关参数层次结构的更多信息，请参阅[将参数组织成层次结构](#) (p. 366)。

使用层次结构管理参数

1. 将 AWS CLI [下载](#)到本地计算机上。
2. 打开 AWS CLI 并运行以下命令指定您的凭证和区域。您必须在 Amazon EC2 中具有管理员权限，或者您必须在 AWS Identity and Access Management (IAM) 中被授予相应权限。

```
aws configure
```

系统将提示您指定以下内容。

```
AWS Access Key ID [None]: key_name
AWS Secret Access Key [None]: key_name
Default region name [None]: region
Default output format [None]: ENTER
```

3. 执行以下命令创建一个使用 `allowedPattern` 参数和 `String` 数据类型的参数。本示例中的允许模式表示参数的值必须为 1 到 4 个数字。

```
aws ssm put-parameter --name "/MyService/Test/MaxConnections" --value 100 --allowed-
pattern "\d{1,4}" --type String
```

该命令返回参数的版本号。

4. 执行以下命令尝试用新值覆盖您刚刚创建的参数。

```
aws ssm put-parameter --name "/MyService/Test/MaxConnections" --value 10,000 --type
String --overwrite
```

系统会引发以下错误，因为新值不满足您在上一步骤中指定的允许模式的要求。

```
An error occurred (ParameterPatternMismatchException) when calling the
PutParameter operation: Parameter value, cannot be validated against
allowedPattern: \d{1,4}
```

5. 执行以下命令创建一个使用您的默认 AWS KMS 密钥的安全字符串参数。本示例中的允许模式表示用户可以指定任意字符，并且值必须在 8 到 20 个字符之间。

```
aws ssm put-parameter --name "/MyService/Test/my-password" --value "p#sW*rd33" --
allowed-pattern ".{8,20}" --type SecureString
```

Important

只有安全字符串参数的值会被加密。参数的名称、描述和其他属性不会被加密。因此，为了让包含密码的参数不那么明显，我们建议使用在参数名称中避免“password”实际字样的命名系统。不过，为了便于说明，我们在示例中使用了 *my-password* 示例参数名称。

6. 执行以下命令创建多个使用上一步骤中的层次结构的参数。

```
aws ssm put-parameter --name "/MyService/Test/DBname" --value "SQLDevDb" --type String
```

```
aws ssm put-parameter --name "/MyService/Test/user" --value "SA" --type String
```

```
aws ssm put-parameter --name "/MyService/Test/userType" --value "SQLuser" --type String
```

7. 执行以下命令获取两个参数的值。

```
aws ssm get-parameters --names "/MyService/Test/user" "/MyService/Test/userType"
```

8. 执行以下命令查询单个级别内的所有参数。

```
aws ssm get-parameters-by-path --path "/MyService/Test"
```

9. 执行以下命令删除两个参数

```
aws ssm delete-parameters --names "/IADRegion/Dev/user" "/IADRegion/Dev/userType"
```

使用 AWS Systems Manager 监控实例

SSM 代理将有关执行、计划操作、错误和运行状况的信息写入每个实例上的日志文件。手动连接到实例以查看日志文件并对 SSM 代理的问题进行排查是耗时的工作。为更有效地监控实例，可以配置 SSM 代理本身或 CloudWatch 代理将此日志数据发送到 Amazon CloudWatch Logs。

Important

统一的 CloudWatch 代理已取代 SSM 代理 作为将日志数据发送到 Amazon CloudWatch Logs 的工具。对使用 SSM 代理 发送日志数据的支持将在不久的将来被弃用。我们建议您尽快开始将统一的 CloudWatch 代理用于日志收集过程。有关更多信息，请参阅以下主题：

- [将日志发送到 CloudWatch Logs \(CloudWatch 代理\) \(p. 390\)](#)
- [将 Windows Server 实例日志收集迁移到 CloudWatch 代理 \(p. 391\)](#)
- Amazon CloudWatch User Guide 中的 [使用 CloudWatch 代理从 Amazon Elastic Compute Cloud 实例和本地服务器中收集指标](#)。

借助 CloudWatch Logs，可以实时监控日志数据，可以通过创建一个或多个指标筛选器来搜索和筛选日志数据，可以存档历史数据并在需要时进行检索。有关 CloudWatch Logs 的更多信息，请参阅 [Amazon CloudWatch Logs User Guide](#)。

配置代理将日志数据发送到 Amazon CloudWatch Logs 有以下好处：

- 您的所有 SSM 代理日志文件的集中日志文件存储。
- 实现对文件的更快访问，以调查错误。
- 无限日志文件保留 (可配置)。
- 无论实例的状态如何，都可以维护和访问日志。
- 对其他 CloudWatch 功能 (例如指标和警报) 的访问。

主题

- [将日志发送到 CloudWatch Logs \(SSM 代理\) \(p. 389\)](#)
- [将日志发送到 CloudWatch Logs \(CloudWatch 代理\) \(p. 390\)](#)
- [使用 AWS CloudTrail 记录 Systems Manager API 调用 \(p. 395\)](#)

将日志发送到 CloudWatch Logs (SSM 代理)

AWS Systems Manager 代理是在您的 Amazon EC2 实例和为 Systems Manager (混合实例) 配置的混合实例上运行的 Amazon 软件。SSM 代理在云中从 Systems Manager 服务处理请求并按照请求中指定的方式配置计算机。有关 SSM 代理的更多信息，请参阅 [安装和配置 SSM 代理 \(p. 14\)](#)。

此外，按照以下步骤，您可以配置 SSM 代理将日志数据发送到 Amazon CloudWatch Logs。

Important

统一的 CloudWatch 代理已取代 SSM 代理 作为将日志数据发送到 Amazon CloudWatch Logs 的工具。对使用 SSM 代理 发送日志数据的支持将在不久的将来被弃用。我们建议您尽快开始将统一的 CloudWatch 代理用于日志收集过程。有关更多信息，请参阅以下主题：

- [将日志发送到 CloudWatch Logs \(CloudWatch 代理\) \(p. 390\)](#)

- 将 Windows Server 实例日志收集迁移到 CloudWatch 代理 (p. 391)
- Amazon CloudWatch User Guide 中的 [使用 CloudWatch 代理从 Amazon Elastic Compute Cloud 实例和本地服务器中收集指标](#)。

开始前的准备工作

在 Amazon CloudWatch Logs 中创建日志组。有关更多信息，请参阅 Amazon CloudWatch Logs User Guide 中的 [在 CloudWatch Logs 中创建日志组](#)。

配置 SSM 代理以将日志发送到 CloudWatch

1. 登录实例并找到以下文件：

在 Windows 上: %PROGRAMFILES%\Amazon\SSM\seelog.xml.template

在 Linux 上: /etc/amazon/ssm/seelog.xml.template

2. 将文件名从 seelog.xml.template 更改为 seelog.xml。
3. 用文本编辑器打开 seelog.xml 文件并找到以下部分：

```
<outputs formatid="fmtinfo">
  <console formatid="fmtinfo"/>
  <rollingfile type="size" maxrolls="5" maxsize="30000000"
filename="{LOCALAPPDATA}\Amazon\SSM\Logs\amazon-ssm-agent.log"/>
  <filter formatid="fmterror" levels="error,critical">
  <rollingfile type="size" maxrolls="5" maxsize="10000000"
filename="{LOCALAPPDATA}\Amazon\SSM\Logs\errors.log"/>
  </filter>
</outputs>
```

4. 编辑文件，并将以下 custom name 元素添加到结尾的 </filter> 标签之后，如下示例中所示。

```
<seelog minlevel="info" critmsgcount="500" maxinterval="100000000"
mininterval="2000000" type="adaptive">
  <exceptions>
    <exception minlevel="error" filepattern="test*" />
  </exceptions>
  <outputs formatid="fmtinfo">
    <console formatid="fmtinfo"/>
    <rollingfile type="size" maxrolls="5" maxsize="30000000"
filename="{LOCALAPPDATA}\Amazon\SSM\Logs\amazon-ssm-agent.log"/>
    <filter formatid="fmterror" levels="error,critical">
    <rollingfile type="size" maxrolls="5" maxsize="10000000"
filename="{LOCALAPPDATA}\Amazon\SSM\Logs\errors.log"/>
    </filter>
    <custom name="cloudwatch_receiver" formatid="fmtdebug" data-log-group="Your
CloudWatch Log Group Name" />
  </outputs>
```

5. 保存您的更改，然后重新启动 SSM 代理或实例。
6. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
7. 选择 Logs，然后选择您的日志组。(SSM 代理 日志文件数据的日志流按实例 ID 进行组织。)

将日志发送到 CloudWatch Logs (CloudWatch 代理)

您可以配置和使用 Amazon CloudWatch 代理从实例收集指标和日志，而不是使用 SSM 代理完成此类任务。CloudWatch 代理能够收集的 Amazon EC2 实例相关指标比 SSM 代理要多。此外，还可以使用 CloudWatch 代理从本地服务器收集指标。

也可以将代理配置设置存储在 Systems Manager Parameter Store 中，以供 CloudWatch 代理使用。

Note

Currently, AWS Systems Manager supports migrating from SSM 代理 to the CloudWatch 代理 for collecting logs and metrics on 64-bit versions of Windows only. For information about setting up the CloudWatch 代理 on other operating systems, and for complete information about using the CloudWatch 代理, see [Collect Metrics from Amazon Elastic Compute Cloud Instances and On-Premises Servers with the CloudWatch 代理](#) in the [Amazon CloudWatch User Guide](#).

You can use the CloudWatch 代理 on other supported operating systems, but you will not be able to use Systems Manager to perform a tool migration.

主题

- [将 Windows Server 实例日志收集迁移到 CloudWatch 代理 \(p. 391\)](#)
- [将 CloudWatch 代理配置设置存储到 Parameter Store 中 \(p. 394\)](#)
- [回滚到使用 SSM 代理进行日志收集 \(p. 394\)](#)

将 Windows Server 实例日志收集迁移到 CloudWatch 代理

如果当前在支持的 Windows Server 实例上使用 SSM 代理将 SSM 代理日志文件发送到 Amazon CloudWatch Logs，则可以使用 Systems Manager 从 SSM 代理迁移到 CloudWatch 代理作为日志收集工具，并迁移配置设置。

CloudWatch 代理在 32 位版本的 Windows Server 上不受支持。

对于 64 位 Amazon EC2 Windows 实例，可以自动或手动执行到 CloudWatch 代理的迁移。对于本地实例，必须手动执行此过程。

Note

迁移期间，发送到 CloudWatch 的数据可能会中断或重复发送。迁移完成后，指标和日志数据会再次准确记录到 CloudWatch 中。

建议先在数量有限的实例上测试迁移，再将整个队列迁移到 CloudWatch 代理。迁移后，如果更倾向于使用 SSM 代理进行日志收集，可以改为使用 SSM 代理。

Important

在以下情况下，不能使用本主题中介绍的步骤向 CloudWatch 代理迁移：

- SSM 代理的现有配置指定多个区域。
- SSM 代理的现有配置指定多组访问/私有密钥凭证。

在上述情况下，必须在 SSM 代理中禁用日志收集并安装 CloudWatch 代理，而不执行迁移过程。有关更多信息，请参阅以下主题：

- [在 Amazon EC2 实例上安装 CloudWatch 代理](#)
- [在本地服务器上安装 CloudWatch 代理](#)

开始前的准备工作

在开始迁移到 CloudWatch 代理进行日志收集前，请确保将对其执行迁移的实例满足以下要求：

- 操作系统为 64 位 Windows Server 版本。
- 实例上安装了 SSM 代理 2.2.93.0 或更高版本。
- 已在实例上配置 SSM 代理进行监控。

主题

- [自动迁移到 CloudWatch 代理 \(p. 392\)](#)
- [手动迁移到 CloudWatch 代理 \(p. 392\)](#)

自动迁移到 CloudWatch 代理

对于仅 Amazon EC2 Windows 实例，可以使用 AWS Systems Manager 控制台或 AWS CLI 自动迁移到 CloudWatch 代理作为日志收集工具。

Note

Currently, AWS Systems Manager supports migrating from SSM 代理 to the CloudWatch 代理 for collecting logs and metrics on 64-bit versions of Windows only. For information about setting up the CloudWatch 代理 on other operating systems, and for complete information about using the CloudWatch 代理, see [Collect Metrics from Amazon Elastic Compute Cloud Instances and On-Premises Servers with the CloudWatch 代理](#) in the [Amazon CloudWatch User Guide](#).

You can use the CloudWatch 代理 on other supported operating systems, but you will not be able to use Systems Manager to perform a tool migration.

迁移成功后，请在 CloudWatch 中检查结果，确保正在接收预期的指标、日志或 Windows 事件日志。如果对结果满意，可以选择[将 CloudWatch 代理配置设置存储到 Parameter Store 中 \(p. 394\)](#)。如果迁移不成功或者结果不符合预期，可以[回滚到使用 SSM 代理进行日志收集 \(p. 394\)](#)。

自动迁移到 CloudWatch 代理 (控制台)

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command，然后选择运行命令。

Note

如果打开了 AWS Systems Manager 主页，请向下滚动并选择浏览 Run Command。

3. 在命令文档列表中，选择 AmazonCloudWatch-MigrateCloudWatchAgent。
4. 在目标部分，选择一个选项，然后选择要更新的实例。
5. 选择 Run。

自动迁移到 CloudWatch 代理 (AWS CLI)

- 运行以下命令：

```
aws ssm send-command --document-name AmazonCloudWatch-MigrateCloudWatchAgent --targets  
Key=instanceids,Values=ID1,ID2,ID3
```

ID1、**ID2** 和 **ID3** 表示要更新的实例的 ID，例如 i-1234567890abcdef0。

手动迁移到 CloudWatch 代理

对于本地 Windows 实例或 Amazon EC2 Windows 实例，请按照以下步骤将日志收集手动迁移到 Amazon CloudWatch 代理。

第一步：安装 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command，然后选择运行命令。

Note

如果打开了 AWS Systems Manager 主页，请向下滚动并选择浏览 Run Command。

3. 在命令文档列表中，选择 AWS-ConfigureAWSPackage。
4. 在目标部分，选择一个选项，然后选择要更新的实例。
5. 在操作列表中，选择##。
6. 在名称中，键入 **AmazonCloudWatchAgent**。
7. 在版本中，键入## (如果未默认提供)。
8. 选择 Run。

第二步：更新配置数据 JSON 格式

- 要更新 CloudWatch 代理现有配置设置的 JSON 格式，请使用 AWS Systems Manager Run Command 或直接使用 RDP 连接登录实例，然后在实例上运行以下 Windows PowerShell 命令 (一次运行一条命令)：

```
cd ${Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent
```

```
.\amazon-cloudwatch-agent-config-wizard.exe --isNonInteractiveWindowsMigration
```

{Env:ProgramFiles} 表示可找到包含 CloudWatch 代理的 Amazon 文件夹的位置，通常为 C:\Program Files。

第三步：配置并启动 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command，然后选择运行命令。

Note

如果打开了 AWS Systems Manager 主页，请向下滚动并选择浏览 Run Command。

3. 在 Command document 列表中，选择 AWS-RunPowerShellScript。
4. 在目标部分，选择一个选项，然后选择要更新的实例。
5. 在命令框中，输入以下两条命令：

```
cd ${Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent
```

```
.\amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c file:config.json -s
```

{Env:ProgramFiles} 表示可找到包含 CloudWatch 代理的 Amazon 文件夹的位置，通常为 C:\Program Files。

6. 选择 Run。

第四步：在 SSM 代理中禁用日志收集

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。

2. 在导航窗格中，选择 Run Command，然后选择运行命令。

Note

如果打开了 AWS Systems Manager 主页，请向下滚动并选择浏览 Run Command。

3. 在命令文档列表中，选择 AWS-ConfigureCloudWatch。
4. 在目标部分，选择一个选项，然后选择要更新的实例。
5. 在状态列表中，选择##。
6. 选择 Run。

完成上述步骤后，请在 CloudWatch 中检查日志，确保正在接收预期的指标、日志或 Windows 事件日志。如果对结果满意，可以选择[将 CloudWatch 代理配置设置存储到 Parameter Store 中 \(p. 394\)](#)。如果迁移不成功或者结果不符合预期，可以[回滚到使用 SSM 代理进行日志收集 \(p. 394\)](#)。

将 CloudWatch 代理配置设置存储到 Parameter Store 中

您可以将 Amazon CloudWatch 代理配置文件的内容存储到 Parameter Store 中。通过将此配置数据保存在一个参数中，多个实例可以从它获取配置设置，您不必再在实例上创建或手动更新配置文件。例如，您可以使用 Run Command 将此参数的内容写入多个实例的配置文件，或使用 状态管理器 帮助避免实例队列间的 CloudWatch 代理配置设置出现配置偏差。

运行 CloudWatch 代理配置向导时，可以选择让向导将配置设置保存为 Parameter Store 中的新参数。有关运行 CloudWatch 代理配置向导的信息，请参阅[使用向导创建 CloudWatch 代理配置文件](#)。

如果在运行向导时没有选择将设置保存为参数的选项，或者手动创建了 CloudWatch 代理配置文件，则可以在以下文件中检索要在实例上保存为参数的数据：

```
${Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent\config.json
```

`{Env:ProgramFiles}` 表示可找到包含 CloudWatch 代理的 Amazon 文件夹的位置，通常为 C:\Program Files。

建议在此实例以外的位置保留此文件中 JSON 内容的备份。

有关创建参数的信息，请参阅[创建 Systems Manager 参数 \(p. 369\)](#)。

有关 CloudWatch 代理的更多信息，请参阅 [Amazon CloudWatch User Guide](#) 中的[使用 CloudWatch 代理从 Amazon Elastic Compute Cloud 实例和本地服务器中收集指标](#)。

回滚到使用 SSM 代理进行日志收集

如果需要恢复为使用 SSM 代理进行日志收集，请执行以下步骤。

第一步：从 SSM 代理检索配置数据

1. 在需要恢复为使用 SSM 代理收集日志的实例上，找到 SSM 代理配置文件的内容。此 JSON 文件通常位于以下位置：

```
${Env:ProgramFiles}\\Amazon\\SSM\\Plugins\\awsCloudWatch\\  
\\AWS.EC2.Windows.CloudWatch.json
```

`{Env:ProgramFiles}` 表示可找到 Amazon 文件夹的位置，通常为 C:\Program Files。

2. 将这些数据复制到一个文本文件中，以便在后面的步骤中使用。

建议在此实例以外的位置存储此 JSON 的备份。

第二步：卸载 CloudWatch 代理

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command，然后选择运行命令。

Note

如果打开了 AWS Systems Manager 主页，请向下滚动并选择浏览 Run Command。

3. 在命令文档列表中，选择 AWS-ConfigureAWSPackage。
4. 在目标部分，选择一个选项，然后选择要更新的实例。
5. 在操作列表中，选择##。
6. 在名称中，键入 **AmazonCloudWatchAgent**。
7. 选择 Run。

第三步：在 SSM 代理中重新启用日志收集

1. 在 <https://console.aws.amazon.com/systems-manager/> 上打开 AWS Systems Manager 控制台。
2. 在导航窗格中，选择 Run Command，然后选择运行命令。

Note

如果打开了 AWS Systems Manager 主页，请向下滚动并选择浏览 Run Command。

3. 在命令文档列表中，选择 AWS-ConfigureCloudWatch。
4. 在目标部分，选择一个选项，然后选择要更新的实例。
5. 在状态列表中，选择###。
6. 在属性框中，将保存的旧配置数据的内容粘贴到文本文件中。
7. 选择 Run。

使用 AWS CloudTrail 记录 Systems Manager API 调用

AWS Systems Manager 与 CloudTrail 集成，后者是一种服务，它在 AWS 账户中捕获由 Systems Manager 自身或代表其发出的 API 调用，并将日志文件传输到您指定的 Amazon S3 存储桶。CloudTrail 从 Systems Manager 控制台、通过 AWS CLI 从 Systems Manager 命令、从 AWS Tools for Windows PowerShell 或从直接调用的 Systems Manager API 捕获 API 调用。通过 CloudTrail 收集的信息，您可以确定向 Systems Manager 发出了什么请求、发出请求的源 IP 地址、何人发出的请求以及发出请求的时间等。要了解有关 CloudTrail 的更多信息，包括如何对其进行配置和启用，请参阅 [AWS CloudTrail User Guide](#)

主题

- [CloudTrail 中的 Systems Manager 信息 \(p. 395\)](#)
- [了解 Systems Manager 日志文件条目 \(p. 396\)](#)

CloudTrail 中的 Systems Manager 信息

在您的 AWS 账户中启用 CloudTrail 日志记录后，将在日志文件中跟踪对 Systems Manager 执行的 API 调用。Systems Manager 记录与其他 AWS 服务记录一起写入日志文件。CloudTrail 基于时间段和文件大小来确定何时创建新文件并向其写入内容。

所有 Systems Manager 操作都会记录，[AWS Systems Manager API Reference](#)、[AWS Systems Manager section of the AWS CLI Command Reference](#) 和 [AWS Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference](#) 中对这些操作进行了介绍。例如，创建、删除和编辑 Maintenance Window 以及创建自定义补丁基准的调用会在 CloudTrail 日志文件中生成条目。

每个日志条目都包含有关生成请求的人员的信息。日志中的用户身份信息有助于确定发出的请求是否具有根或 IAM 用户证书，是否具有角色或联合用户的临时安全证书，或者是否是由其他 AWS 服务发出的。有关更多信息，请参阅 [CloudTrail 事件参考](#) 中的 `userIdentity` 字段。

日志文件可以在存储桶中存储任意长时间，不过您也可以定义 Amazon S3 生命周期规则以自动存档或删除日志文件。默认情况下，将使用 Amazon S3 服务器端加密 (SSE) 对日志文件进行加密。

如果需要针对日志文件传输快速采取措施，可选择让 CloudTrail 在传输新日志文件时发布 Amazon SNS 通知。有关更多信息，请参阅 [AWS CloudTrail User Guide](#) 中的 [配置 Amazon SNS 通知](#)。

您也可以将多个 AWS 区域和多个 AWS 账户中的 Systems Manager 日志文件聚合到单个 Amazon S3 存储桶中。有关更多信息，请参阅 [AWS CloudTrail User Guide](#) 中的 [将 CloudTrail 日志文件聚合到单个 Amazon S3 存储桶中](#)。

了解 Systems Manager 日志文件条目

CloudTrail 日志文件可包含一个或多个日志条目，每个条目由多个 JSON 格式的事件组成。一个日志条目表示来自任何源的一个请求，包括有关所请求的操作、所有参数以及操作的日期和时间等信息。不能保证日志条目具有任何特定顺序 (即，它们不是公共 API 调用的有序堆栈跟踪)。

以下示例显示了一个 CloudTrail 日志条目，演示对 US East (Ohio) Region (us-east-2) 中名为 `example-Document` 的文档执行的 Systems Manager 删除文档操作：

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAI44QH8DHBEXAMPLE:203.0.113.11",
    "arn": "arn:aws:sts::123456789012:assumed-role/example-role/203.0.113.11",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-03-06T20:19:16Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/example-role",
        "accountId": "123456789012",
        "userName": "example-role"
      }
    }
  },
  "eventTime": "2018-03-06T20:30:12Z",
  "eventSource": "ssm.amazonaws.com",
  "eventName": "DeleteDocument",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.11",
  "userAgent": "example-user-agent-string",
  "requestParameters": {
    "name": "example-Document"
  },
  "responseElements": null,
  "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
  "eventID": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
}
```

```
"resources": [  
  {  
    "ARN": "arn:aws:ssm:us-east-2:123456789012:document/example-Document",  
    "accountId": "123456789012"  
  },  
],  
"eventType": "AwsApiCall",  
"recipientAccountId": "123456789012"  
}
```

AWS Systems Manager 的身份验证和访问控制

访问 AWS Systems Manager 需要凭证。这些凭证必须有权访问 AWS 资源，以执行创建或更新文档、向 Maintenance Window 注册任务和目标等任务。以下部分详细介绍如何使用 [AWS Identity and Access Management \(IAM\)](#) 和 Systems Manager 帮助保护对您的资源的访问。

- [身份验证 \(p. 398\)](#)
- [访问控制 \(p. 399\)](#)

有关配置 AWS Systems Manager 访问权限的更多信息，请参阅[配置对 Systems Manager 的访问权限 \(p. 9\)](#)。

有关资源为执行 Systems Manager 操作而可能需要访问的 Amazon Simple Storage Service (Amazon S3) 存储桶的信息，请参阅[SSM 代理的最低 S3 存储桶权限 \(p. 29\)](#)。

身份验证

您可以以下面任一类型的身份访问 AWS：

- **AWS 账户根用户** – 注册 AWS 时，您需要提供与您的 AWS 账户关联的电子邮件地址和密码。这些是您的根凭证，它们提供对您所有 AWS 资源的完全访问权限。

Important

出于安全考虑，我们建议您仅使用根凭证创建管理员用户，此类用户是对您的 AWS 账户具有完全访问权限的 IAM 用户。随后，您可以使用此管理员用户来创建具有有限权限的其他 IAM 用户和角色。有关更多信息，请参阅 IAM User Guide 中的 [IAM 最佳实践](#)和[创建管理员用户和组](#)。

- **IAM 用户** - [IAM 用户](#)就是您的 AWS 账户中的一种身份，它具有特定的自定义权限（例如，用于在 Systems Manager 中向目标发送事件数据的权限）。您可以使用 IAM 用户名和密码来登录以保护 AWS 网页，如 [AWS Management Console](#)、[AWS 开发论坛](#)或 [AWS Support Center](#)。

除了用户名和密码之外，您还可以为每个用户生成[访问密钥](#)。在通过[多个软件开发工具包之一](#)或使用 [AWS Command Line Interface \(AWS CLI\)](#) 以编程方式访问 AWS 服务时，用户可以使用这些密钥。SDK 和 CLI 工具使用访问密钥对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。Systems Manager supports 签名版本 4，后者是一种用于对入站 API 请求进行身份验证的协议。有关验证请求的更多信息，请参阅 AWS General Reference 中的[签名版本 4 签名流程](#)。

- **IAM 角色** - [IAM 角色](#)是可在账户中创建的另一种具有特定权限的 IAM 身份。它类似于 IAM 用户，但未与特定人员关联。利用 IAM 角色，您可以获得可用于访问 AWS 服务和资源的临时访问密钥。具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合身份用户访问 – 您可以不创建 IAM 用户，而是使用来自 AWS Directory Service、您的企业用户目录或 Web 身份提供商的既有用户身份。他们被称为联合身份用户。在通过[身份提供商](#)请求访问权限时，AWS 将为联合用户分配角色。有关联合身份用户的更多信息，请参阅 IAM User Guide 中的[联合身份用户和角色](#)。
- 跨账户访问 – 可以使用您账户中的 IAM 角色向另一个 AWS 账户授予对您账户的资源的访问权限。有关示例，请参阅 IAM User Guide 中的[教程：使用 IAM 角色委派跨 AWS 账户的访问权限](#)。
- AWS 服务访问 – 可以使用您账户中的 IAM 角色向 AWS 服务授予对您账户的资源的访问权限。例如，您可以创建一个角色，此角色允许 Amazon Redshift 代表您访问 Amazon S3 存储桶，然后将存储在该存储桶中的数据加载到 Amazon Redshift 群集中。有关更多信息，请参阅 IAM User Guide 中的[创建向 AWS 服务委派权限的角色](#)。
- 在 Amazon EC2 上运行的应用程序 – 您不用将访问密钥存储在 EC2 实例中以供实例上运行的应用程序使用并发出 AWS API 请求，而是可以使用 IAM 角色管理这些应用程序的临时凭证。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM User Guide 中的[对 Amazon EC2 上的应用程序使用角色](#)。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 Systems Manager 资源。例如，您必须有权创建、查看或删除激活、关联、文档，Maintenance Window 必须有权注册或取消注册实例和补丁基准等。

下面几节介绍如何管理 Systems Manager 的权限。我们建议您先阅读概述。

- [管理您的 AWS Systems Manager 资源的访问权限概述 \(p. 399\)](#)
- [为 AWS Systems Manager 使用基于身份的策略 \(IAM 策略\) \(p. 404\)](#)
- [AWS Systems Manager 权限参考 \(p. 408\)](#)

管理您的 AWS Systems Manager 资源的访问权限概述

每个 AWS 资源都归某个 AWS 账户所有，创建和访问资源的权限由权限策略进行管理。账户管理员可以向 IAM 身份（即：用户、组和角色）挂载权限策略。AWS Lambda、Amazon Simple Notification Service (Amazon SNS)、Amazon Simple Storage Service (Amazon S3) 等服务也支持向资源附加权限策略。

Note

账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅 IAM User Guide 中的[IAM 最佳实践](#)。

在授予权限时，账户管理员决定谁获得权限，获得对哪些资源的权限，以及您允许对这些资源执行的具体操作。

主题

- [AWS Systems Manager 资源和操作 \(p. 400\)](#)

- [了解资源所有权 \(p. 402\)](#)
- [管理对资源的访问 \(p. 402\)](#)
- [指定策略元素：资源、操作、效果和委托人 \(p. 403\)](#)
- [在策略中指定条件 \(p. 404\)](#)

AWS Systems Manager 资源和操作

Systems Manager 包含几种主要资源：

- 自动化定义
- 自动化执行
- 文档
- Maintenance Window
- 托管实例
- 托管实例清单
- 参数
- 补丁基准
- 资源数据同步

对于自动化定义，Systems Manager 支持二级资源版本 ID。在 AWS 中，这些二级资源称作子资源。如果指定自动化定义资源的版本子资源，您就可以访问自动化定义的特定版本。例如，您可能需要确保在实例管理中只使用最新版本的自动化定义。

要组织和管理参数，可以使用分层结构为参数创建名称。在分层结构中，参数名称可以包含使用正斜杠定义的路径。您可以命名最多包含五个级别的参数资源。建议创建反映环境中现有层次结构的层次结构。有关更多信息，请参阅 [创建 Systems Manager 参数 \(p. 369\)](#)。

每个资源都有唯一的 Amazon 资源名称 (ARN)。在策略中，使用 ARN 识别策略要应用到的资源。有关 ARN 的详细信息，请参阅 Amazon Web Services General Reference 中的 [Amazon 资源名称 \(ARN\)](#) 和 [AWS 服务命名空间](#)。

下表列出了 Systems Manager 中每个资源类型的 ARN 格式结构：

资源类型	ARN 格式
自动化执行	arn:aws:ssm: <i>region</i> : <i>account-id</i> :automation-execution/ <i>automation-execution-id</i>
自动化定义 (使用版本子资源)	arn:aws:ssm: <i>region</i> : <i>account-id</i> :automation-definition/ <i>automation-definition-id:version-id</i>
文档	arn:aws:ssm: <i>region</i> : <i>account-id</i> :document/ <i>document-name</i>
Maintenance Window	arn:aws:ssm: <i>region</i> : <i>account-id</i> :maintenancewindow/ <i>window-execution-id</i>
Maintenance Window 任务	arn:aws:ssm: <i>region</i> : <i>account-id</i> :windowtask/ <i>window-task-id</i>
Maintenance Window 目标	arn:aws:ssm: <i>region</i> : <i>account-id</i> :windowtarget/ <i>window-target-id</i>
托管实例	arn:aws:ssm: <i>region</i> : <i>account-id</i> :managed-instance/ <i>managed-instance-id</i>

资源类型	ARN 格式
托管实例清单	arn:aws:ssm: <i>region</i> : <i>account-id</i> :managed-instance-inventory/ <i>managed-instance-id</i>
参数	一级参数： <ul style="list-style-type: none">arn:aws:ssm:<i>region</i>:<i>account-id</i>:parameter/<i>parameter-name</i>/ 使用分层结构命名的参数： <ul style="list-style-type: none">arn:aws:ssm:<i>region</i>:<i>account-id</i>:parameter/<i>parameter-name-root/level-2/level-3/level-4/level-5</i>
补丁基准	arn:aws:ssm: <i>region</i> : <i>account-id</i> :patchbaseline/ <i>patch-baseline-id</i>
所有 Systems Manager 资源	arn:aws:ssm:*
指定账户在指定区域拥有的所有 Systems Manager 资源	arn:aws:ssm: <i>region</i> : <i>account-id</i> :*

Note

大多数 AWS 服务将 ARN 中的冒号 (:) 或正斜杠 (/) 视为同一个字符。不过，Systems Manager 在资源模式和规则中要求精确匹配。创建事件模式时，请务必使用正确的 ARN 字符，以使其与资源的 ARN 匹配。

例如，您可以使用某个特定文档 (*myDocument*) 的 ARN 在语句中指定该规则，如下所示：

```
"Resource": "arn:aws:ssm:us-west-2:123456789012:document/myDocument"
```

也可以使用通配符 (*) 指定属于特定账户的所有 documents，如下所示：

```
"Resource": "arn:aws:ssm:us-west-2:123456789012:document/*"
```

对于 Parameter Store API 操作，可以使用层次结构名称和 AWS Identity and Access Management (IAM) 策略提供或限制对层次结构的一个级别中所有参数的访问权限，如下所示：

```
"Resource": "arn:aws:ssm:us-west-2:123456789012:parameter/Dev/ERP/Oracle/*"
```

要指定所有资源，或者，如果特定 API 操作不支持 ARN，请在 Resource 元素中使用通配符 (*)，如下所示：

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

有些 Systems Manager API 操作接受多个资源。要在单个语句中指定多种资源，请使用逗号将它们隔开，如下所示：

```
"Resource": ["arn1", "arn2"]
```

有关可用于这些资源类型的 Systems Manager 操作的列表，请参阅 [AWS Systems Manager 权限参考 \(p. 408\)](#)。

了解资源所有权

资源所有者 是创建资源的 AWS 账户，与资源具体是由此账户中的谁创建的无关。具体而言，资源所有者是对资源创建请求进行身份验证的 [委托人实体](#) (根账户、IAM 用户或 IAM 角色) 的 AWS 账户。以下示例说明了它的工作原理：

- 如果您使用 AWS 账户的根账户凭证创建规则，则您的 AWS 账户即为该 Systems Manager 资源的所有者。
- 如果您在您的 AWS 账户中创建 IAM 用户并对该用户授予创建 Systems Manager 资源的权限，则该用户可以创建 Systems Manager 资源。但是，该用户所属的 AWS 账户拥有这些 Systems Manager 资源。
- 如果您在您的 AWS 账户中创建具有创建 Systems Manager 资源的权限的 IAM 角色，则能够担任该角色的任何人都可以创建 Systems Manager 资源。该角色所属的 AWS 账户拥有这些 Systems Manager 资源。

管理对资源的访问

权限策略 规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

Note

本节讨论如何在 Systems Manager 范围内使用 IAM。它不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅 [什么是 IAM？](#) (在 IAM User Guide 中)。有关 IAM 策略语法和说明的信息，请参阅 IAM User Guide 中的 [AWS IAM 策略参考](#)。

附加到 IAM 身份的策略称作基于身份的策略 (IAM 策略)。附加到资源的策略称作基于资源的策略。Systems Manager 只支持基于身份的策略。

主题

- [基于身份的策略 \(IAM 策略\) \(p. 402\)](#)
- [基于资源的策略 \(p. 403\)](#)

基于身份的策略 (IAM 策略)

您可以向 IAM 身份挂载策略。通过创建基于身份的 IAM 策略，您可以限制账户中的用户有权访问的调用和资源，然后将这些策略与 IAM 用户关联。有关如何创建 IAM 角色和探究 Systems Manager 的 IAM 策略语句示例的更多信息，请参阅 [管理您的 AWS Systems Manager 资源的访问权限概述 \(p. 399\)](#)。例如，您可以执行以下操作：

- 将权限策略附加到您的账户中的用户或组 - 要授予查看 AWS Systems Manager 控制台中的应用程序、部署组和其他 Systems Manager 资源的用户权限，您可以将权限策略附加到用户或用户所属的组。
- 向角色挂载权限策略 (授予跨账户权限) - 要授予跨账户的权限，可以向 IAM 角色挂载基于身份的权限策略。例如，账户 A 中的管理员可以创建一个角色，以向其他 AWS 账户 (如账户 B) 或某项 AWS 服务授予跨账户权限，如下所述：

1. 账户 A 管理员创建一个 IAM 角色，向该角色挂载授权其访问账户 A 中资源的权限策略。
2. 账户 A 管理员可以向将账户 B 标识为能够担任该角色的委托人的角色挂载信任策略。
3. 之后，账户 B 管理员可以委派权限，指派账户 B 中的任何用户代入该角色。这样，账户 B 中的用户可以创建或访问账户 A 中的资源。如果需要授予 AWS 服务权限来代入该角色，则信任策略中的委托人也可以是 AWS 服务委托人。

有关使用 IAM 委派权限的更多信息，请参阅 IAM User Guide 中的[访问权限管理](#)。

基于资源的策略对操作类型的访问控制能力因资源类型而异，如下表所述：

资源类型	操作类型
全部	查看和列出有关资源的详细信息
自动化定义	Start Stop
文档	创建
Maintenance Window	删除
参数	更新
托管实例	取消注册 注册
托管实例清单	Create 更新
补丁基准	创建 删除 取消注册 注册 更新
资源数据同步	创建 删除

基于资源的策略

其他 AWS 服务 (如 Amazon Simple Storage Service) 还支持基于资源的权限策略。例如，您可以将权限策略挂载到 S3 存储桶以管理对该存储桶的访问权限。Systems Manager 不支持基于资源的策略。

指定策略元素：资源、操作、效果和委托人

Systems Manager 为每个 Systems Manager 资源都定义一组适用的 API 操作。为允许您授予这些 API 操作的权限，Systems Manager 定义了一组您可以在策略中指定的操作。某些 API 操作可能需要执行多个操作的权限。有关资源和 API 操作的更多信息，请参阅[AWS Systems Manager 资源和操作 \(p. 400\)](#) 和 [AWS Systems Manager 权限参考 \(p. 408\)](#)。有关操作的列表，请参阅[AWS Systems Manager 资源和操作 \(p. 400\)操作](#)。

以下是基本的策略元素：

- **Resource** - 您使用 Amazon 资源名称 (ARN) 来标识策略应用到的资源。对于 Systems Manager 资源，您可以在 IAM 策略中使用通配符 (*)。有关更多信息，请参阅 [AWS Systems Manager 资源和操作 \(p. 400\)](#)。
- **Action** - 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，`ssm:GetDocument` 权限允许执行 `GetDocument` 操作的用户权限。
- **效果** - 您可以指定当用户请求特定操作 (可以是允许或拒绝) 时出现的效果。如果没有显式授予 (允许) 对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。
- **Principal** - 在基于身份的策略 (IAM 策略) 中，附加了策略的用户是隐式委托人。对于基于资源的策略，您可以指定您希望将权限授予的用户、账户、服务或其他实体。Systems Manager 只支持基于身份的策略。

要了解有关 IAM 策略语法和说明的更多信息，请参阅 IAM User Guide 中的 [AWS IAM 策略参考](#)。

有关显示所有 Systems Manager API 操作及其适用资源的表，请参阅 [AWS Systems Manager 权限参考 \(p. 408\)](#)。

在策略中指定条件

当您授予权限时，可使用访问策略中的语言指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅 IAM User Guide 中的 [条件](#)。

要表示条件，您可以使用预定义的条件键。没有特定于 AWS Systems Manager 的条件键。但有 AWS 范围内的条件密钥，您可以根据需要使用。有关 AWS 范围内的键的完整列表，请参阅 IAM User Guide 中的 [条件的可用键](#)。

为 AWS Systems Manager 使用基于身份的策略 (IAM 策略)

下面基于身份的策略示例说明账户管理员如何将权限策略附加到 IAM 身份 (即用户、组和角色)，从而授予对 Systems Manager 资源执行操作的权限。

Important

我们建议您首先阅读以下介绍性主题，这些主题讲解了管理 Systems Manager 资源访问的基本概念和选项。有关更多信息，请参阅 [管理您的 AWS Systems Manager 资源的访问权限概述 \(p. 399\)](#)。

主题

- [使用 AWS Systems Manager 控制台所需的权限 \(p. 405\)](#)
- [适用于 AWS Systems Manager 的 AWS 托管 \(预定义\) 策略 \(p. 405\)](#)
- [客户托管策略示例 \(p. 406\)](#)

以下是一个权限策略示例，允许用户删除 `us-west-2` 区域中名称以 `MyDocument-` 开头的文档。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:DeleteDocument"
      ],

```

```
"Resource" : [
  "arn:aws:ssm:us-west-2:123456789012:document:MyDocument-*"
]
}
]
}co
```

使用 AWS Systems Manager 控制台所需的权限

要使用 AWS Systems Manager 控制台，用户必须拥有一组最低的权限，这样该用户才能在自己的 AWS 账户中描述其他 AWS 资源。要充分使用 Systems Manager 控制台中的 Systems Manager，您必须拥有来自以下服务的权限：

- AWS Systems Manager
- Amazon Elastic Compute Cloud (Amazon EC2)
- AWS Identity and Access Management (IAM)

可以使用以下策略声明授予所需权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:*",
        "ec2:describeInstances",
        "iam:PassRole",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```

如果创建比必需的最低权限更为严格的 IAM 策略，对于附加了该 IAM 策略的用户，控制台将无法按预期正常运行。为确保这些用户能够使用 Systems Manager 控制台，同时向用户附加 AmazonSSMReadOnlyAccess 托管策略，请参阅[适用于 AWS Systems Manager 的 AWS 托管（预定义）策略](#) (p. 405)。

对于只需要调用 AWS CLI 或 Systems Manager API 的用户，您无需为其提供最低控制台权限。

适用于 AWS Systems Manager 的 AWS 托管（预定义）策略

AWS 通过提供由 AWS 创建和管理的独立 IAM 策略来解决许多常用案例。这些 AWS 托管策略 授予常用案例的必要权限，这样您不必调查需要哪些权限。有关更多信息，请参阅 IAM User Guide 中的 [AWS 托管策略](#)。

以下 AWS 托管策略 (可附加到账户中的用户) 特定于 AWS Systems Manager：

- AmazonSSMFullAccess - 授予 Systems Manager API 和文档完全访问权限的用户信任策略。
- AmazonSSMAutomationRole - 该服务角色为 AWS Systems Manager 自动化服务提供权限以运行在自动化文档中定义的活动。将此策略分配给管理员和可信高级用户。

- AmazonSSMReadOnlyAccess - 授予 Systems Manager 只读 API 操作 (例如 `Get*` 和 `List*`) 访问权限的用户信任策略。
- AmazonSSMMaintenanceWindowRole - Systems Manager Maintenance Window 的服务角色。
- AmazonEC2RoleforSSM - 使实例能够与 Systems Manager API 进行通信的实例信任策略。

此外，您还可以创建自己的自定义 IAM 策略，以授予 Systems Manager 操作和资源的相关权限。您可以将这些自定义策略挂载到需要这些权限的 IAM 用户或组。

Note

在混合环境中，您需要一个额外的 IAM 角色，以允许服务器和虚拟机与 Systems Manager 服务进行通信。这就是 Systems Manager 的 IAM 服务角色。此角色向 AWS Security Token Service (AWS STS) AssumeRole 授予对 Systems Manager 服务的信任。AssumeRole 操作返回一组临时安全凭证 (由访问密钥 ID、秘密访问密钥和安全令牌组成)。您使用这些临时凭证访问通常可能无法访问的 AWS 资源。有关更多信息，请参阅 [AWS Security Token Service API Reference](#) 中的 [为混合环境创建 IAM 服务角色 \(p. 31\)](#) 和 [AssumeRole](#)。

客户托管策略示例

您可以创建在自己的 AWS 账户中管理的独立策略。我们将它们称作客户托管策略。您可以将这些策略附加到 AWS 账户中的多个委托人实体。将策略附加到委托人实体时，便向实体授予了策略中定义的权限。有关更多信息，请参阅 [IAM User Guide](#) 中的 [客户托管策略](#)。

以下用户策略示例授予执行各种 AWS Systems Manager 操作的权限。可以使用它们限制 IAM 用户和角色的 Systems Manager 访问。在 Systems Manager API、AWS 软件开发工具包或 AWS CLI 中执行操作时，这些策略将会发挥作用。对于使用控制台的用户，需要授予特定于控制台的其他权限。有关更多信息，请参阅 [使用 AWS Systems Manager 控制台所需的权限 \(p. 405\)](#)。

Note

所有示例都使用 US West (Oregon) Region (us-west-2) 和虚构的账户 ID。

示例

- [示例 1：允许用户在单个区域中执行 Systems Manager 操作 \(p. 406\)](#)
- [示例 2：允许用户列出某个区域的文档 \(p. 407\)](#)

示例 1：允许用户在单个区域中执行 Systems Manager 操作

以下示例授予仅在 **us-west-2** 区域执行 AWS Systems Manager 操作的权限：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "arn:aws:ssm:*"
      ],
      "Resource" : [
        "arn:aws::aws:ssm:us-west-2:111222333444:*"
      ]
    }
  ]
}
```



```
]
}
```

示例 2：允许用户列出某个区域的文档

以下示例授予列出 **us-west-2** 区域中所有以 **Update** 开头的文档名称的权限：

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "ssm:ListDocuments"
      ],
      "Resource" : [
        "arn:aws:ssm:us-west-2:111222333444:document/Update*"
      ]
    }
  ]
}
```

为 Systems Manager 使用服务相关角色

AWS Systems Manager 使用 AWS Identity and Access Management (IAM) [服务相关角色](#)。服务相关角色是一种与 Systems Manager 直接关联的独特类型的 IAM 角色。服务相关角色由 Systems Manager 预定义，具有服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地了解 Systems Manager，因为您不必手动添加必要的权限。Systems Manager 定义其服务相关角色的权限，除非另外定义，否则只有 Systems Manager 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

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 列为 Yes 的服务。选择 Yes 与查看该服务的服务相关角色文档的链接。

用于 Systems Manager 的服务相关角色权限

Systems Manager 使用名为 AWSServiceRoleForAmazonSSM 的服务相关角色 – AWS Systems Manager uses this IAM service role to manage AWS resources on your behalf.

AWSServiceRoleForAmazonSSM 服务相关角色仅信任 ssm.amazonaws.com 服务代入此角色。

仅 Systems Manager Inventory 需要服务相关角色。该角色支持系统收集标签和资源组中的清单元数据。

AWSServiceRoleForAmazonSSM 服务相关角色权限策略允许 Systems Manager 对所有相关资源完成以下操作：

- ssm:CancelCommand
- ssm:GetCommandInvocation
- ssm:ListCommandInvocations
- ssm:ListCommands
- ssm:SendCommand
- ec2:DescribeInstanceAttribute
- ec2:DescribeInstanceStatus

- `ec2:DescribeInstances`
- `resource-groups:ListGroup`
- `resource-groups:ListGroupResources`
- `tag:GetResources`

为 Systems Manager 创建服务相关角色

您可以使用 IAM 控制台用 AWS Service Role for AWS Systems Manager 使用案例创建服务相关角色。在 IAM CLI 或 IAM API 中，用 `ssm.amazonaws.com` 服务名称创建一个服务相关角色。有关更多信息，请参阅 IAM User Guide 中的[创建服务相关角色](#)。如果您删除了此服务相关角色，则可以使用此相同过程再次创建角色。

编辑用于 Systems Manager 的服务相关角色

Systems Manager 不允许您编辑 `AWSServiceRoleForAmazonSSM` 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅 IAM User Guide 中的[编辑服务相关角色](#)。

删除用于 Systems Manager 的服务相关角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。如果您删除了 Systems Manager Inventory 使用的服务相关角色，标签和资源组的清单数据将不再同步。您必须先清除服务相关角色的资源，然后才能手动删除它。

Note

如果在您试图删除标签或资源组时 Systems Manager 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，则请等待几分钟后重试。

删除由 `AWSServiceRoleForAmazonSSM` 使用的 Systems Manager 资源

1. 要删除标签，请参阅[为单个资源添加和删除标签](#)。
2. 要删除资源组，请参阅[从 AWS 资源组删除组](#)。

使用 IAM 手动删除服务相关角色

可以使用 IAM 控制台、IAM CLI 或 IAM API 删除 `AWSServiceRoleForAmazonSSM` 服务相关角色。有关更多信息，请参阅 IAM User Guide 中的[删除服务相关角色](#)。

AWS Systems Manager 权限参考

下表列出了 AWS Systems Manager API 操作及您可授予权限的相应操作。在设置[访问控制](#) (p. 399) 和编写可附加到 IAM 身份的权限策略 (基于身份的策略) 时，可以使用此表作为参考。请在策略的 `Action` 字段中指定这些操作。要指定操作，请在 API 操作名称之前使用 `ssm:` 前缀 (例如，`ssm:GetDocument` 和 `ssm:CreateDocument`)。要在单个语句中指定多项操作，请使用逗号将它们隔开 (例如，`"Action": ["ssm:action1", "ssm:action2"]`)。对于策略的 `Resource` 字段中的资源值，指定一个 ARN。要指定多个操作或资源，可以在 ARN 中使用通配符 (*)。例如，`ssm:*` 指定所有 Systems Manager 操作，`ssm:Get*` 指定以单词 `Get` 开头的所有 Systems Manager 操作。以下示例授予对名称以 `west` 开头的所有文档的访问权限：

```
arn:aws:ssm:us-west-2:111222333444:document:West*
```

有关通配符的更多信息，请参阅 IAM User Guide 中的 [IAM 标识符](#)。有关 ARN 格式的 Systems Manager 资源列表，请参阅 [AWS Systems Manager 资源和操作 \(p. 400\)](#)。

要表示条件，可以在 Systems Manager 策略中使用 AWS 范围条件键。有关 AWS 范围内的密钥的完整列表，请参阅 IAM User Guide 中的 [可用密钥](#)。

指定多个操作或资源

Systems Manager API 操作和必需的操作权限

AddTagsToResource

操作 : `ssm:AddTagsToResource`

添加或覆盖指定资源标签所必需的。

CancelCommand

操作 : `ssm:CancelCommand`

尝试基于指定命令 ID 取消命令所必需的。

CreateActivation

操作 : `ssm:CreateActivation`

向 Amazon EC2 注册本地服务器或虚拟机以便由后者管理所必需的。

CreateAssociation

操作 : `ssm:CreateAssociation`

将 Systems Manager 文档关联至指定实例或目标所必需的。

CreateAssociationBatch

操作 : `ssm:CreateAssociationBatch`

将多个 Systems Manager 文档关联至指定实例或目标所必需的。

CreateDocument

操作 : `ssm:CreateDocument`

创建 Systems Manager 文档所必需的。

CreateMaintenanceWindow

操作 : `ssm:CreateMaintenanceWindow`

创建 Maintenance Window 所必需的。

CreatePatchBaseline

操作 : `ssm:CreatePatchBaseline`

创建补丁基准所必需的。

CreateResourceDataSync

操作 : `ssm:CreateResourceDataSync`

为单个 Amazon S3 存储桶创建资源数据同步配置所必需的。

DeleteActivation

操作 : `ssm>DeleteActivation`

删除激活所必需的。

DeleteAssociation

操作 : `ssm:DeleteAssociation`

从指定实例取消关联指定 Systems Manager 文档所必需的。

DeleteDocument

操作 : `ssm:DeleteDocument`

删除 Systems Manager 文档及此文档的所有实例关联所必需的。

DeleteInventory

操作 : `ssm:DeleteInventory`

删除自定义清单类型或者与自定义清单类型关联的数据所必需的。

DeleteMaintenanceWindow

操作 : `ssm:DeleteMaintenanceWindow`

删除 Maintenance Window 所必需的。

DeleteParameter

操作 : `ssm:DeleteParameter`

从系统中删除参数所必需的。

DeleteParameters

操作 : `ssm:DeleteParameters`

从系统中删除一个或多个参数所必需的。

DeletePatchBaseline

操作 : `ssm:DeletePatchBaseline`

删除补丁基准所必需的。

DeleteResourceDataSync

操作 : `ssm:DeleteResourceDataSync`

删除资源数据同步配置所必需的。

DeregisterManagedInstance

操作 : `ssm:DeregisterManagedInstance`

从注册服务器列表中删除服务器或虚拟机所必需的。

DeregisterPatchBaselineForPatchGroup

操作 : `ssm:DeregisterPatchBaselineForPatchGroup`

从补丁基准中删除补丁组所必需的。

DeregisterTargetFromMaintenanceWindow

操作 : `ssm:DeregisterTargetFromMaintenanceWindow`

从 Maintenance Window 中删除目标所必需的。

DeregisterTaskFromMaintenanceWindow

操作 : `ssm:DeregisterTaskFromMaintenanceWindow`

从 Maintenance Window 中删除任务所必需的。

[DescribeActivations](#)

操作 : `ssm:DescribeActivations`

查看有关激活的详细信息 (如激活的创建日期和时间、到期日期和激活中分配给实例的 IAM 角色) 所必需的。

[DescribeAssociation](#)

操作 : `ssm:DescribeAssociation`

查看指定 Systems Manager 文档或实例的关联所必需的。

[DescribeAssociationExecutions](#)

操作 : `ssm:DescribeAssociationExecutions`

查看特定关联 ID 的所有执行所必需的。

[DescribeAutomationExecutions](#)

操作 : `ssm:DescribeAutomationExecutions`

查看所有激活和已终止的自动化执行信息所必需的。

[DescribeAutomationStepExecutions](#)

操作 : `ssm:DescribeAutomationStepExecutions`

查看自动化流程中所有激活和已终止的步骤执行信息所必需的。

[DescribeAvailablePatches](#)

操作 : `ssm:DescribeAvailablePatches`

查看可能包含在补丁基准中的补丁的信息所必需的。

[DescribeDocument](#)

操作 : `ssm:DescribeDocument`

查看指定 Systems Manager 文档信息所必需的。

[DescribeDocumentPermission](#)

操作 : `ssm:DescribeDocumentPermission`

查看 Systems Manager 文档权限所必需的。

[DescribeEffectiveInstanceAssociations](#)

操作 : `ssm:DescribeEffectiveInstanceAssociations`

查看一个或多个实例的关联的信息所必需的。

[DescribeEffectivePatchesForPatchBaseline](#)

操作 : `ssm:DescribeEffectivePatchesForPatchBaseline`

查看指定补丁基准的当前有效补丁 (补丁和审核状态) 信息所必需的。仅适用于 Windows Server 补丁基准。

[DescribeInstanceAssociationsStatus](#)

操作 : `ssm:DescribeInstanceAssociationsStatus`

查看一个或多个实例的关联的状态所必需的。

DescribeInstanceInformation

操作 : `ssm:DescribeInstanceInformation`

查看一个或多个实例的信息所必需的。

DescribeInstancePatches

操作 : `ssm:DescribeInstancePatches`

查看指定实例的补丁信息及其相对于用于此实例的补丁基准的状态所必需的。

DescribeInstancePatchStates

操作 : `ssm:DescribeInstancePatchStates`

查看一个或多个实例高级补丁状态的信息所必需的。

DescribeInstancePatchStatesForPatchGroup

操作 : `ssm:DescribeInstancePatchStatesForPatchGroup`

查看指定补丁组中实例的高级补丁状态所必需的。

DescribeInventoryDeletions

操作 : `ssm:DescribeInventoryDeletions`

描述特定删除清单操作所必需的。

DescribeMaintenanceWindowExecutions

操作 : `ssm:DescribeMaintenanceWindowExecutions`

查看 Maintenance Window 执行信息所必需的。这包括 Maintenance Window 计划于何时激活的详细信息，以及已向维护窗口注册并由后者运行的任务的相关信息。

DescribeMaintenanceWindowExecutionTaskInvocations

操作 : `ssm:DescribeMaintenanceWindowExecutionTaskInvocations`

检索在 Maintenance Window 执行过程中运行的特定任务的各个任务执行（每个目标一个）的信息所必需的。

DescribeMaintenanceWindowExecutionTasks

操作 : `ssm:DescribeMaintenanceWindowExecutionTasks`

查看为指定 Maintenance Window 执行运行的任务的信息所必需的。

DescribeMaintenanceWindows

操作 : `ssm:DescribeMaintenanceWindows`

查看在 AWS 账户中创建的 Maintenance Window 的信息所必需的。

DescribeMaintenanceWindowTargets

操作 : `ssm:DescribeMaintenanceWindowTargets`

查看向指定 Maintenance Window 注册的目标的信息所必需的。

DescribeMaintenanceWindowTasks

操作 : `ssm:DescribeMaintenanceWindowTasks`

查看指定维护窗口中任务的信息所必需的。

DescribeParameters

操作 : `ssm:DescribeParameters`

查看一个或多个参数的信息所必需的。

DescribePatchBaselines

操作 : `ssm:DescribePatchBaselines`

查看 AWS 账户中的补丁基准的信息所必需的。

DescribePatchGroups

操作 : `ssm:DescribePatchGroups`

查看向补丁基准注册的所有补丁组的信息所必需的。

DescribePatchGroupState

操作 : `ssm:DescribePatchGroupState`

查看补丁组的高级聚合补丁合规性状态的信息所必需的。

GetAutomationExecution

操作 : `ssm:GetAutomationExecution`

查看特定自动化执行详细信息所必需的。

GetCommandInvocation

操作 : `ssm:GetCommandInvocation`

查看调用或插件的命令执行详细信息所必需的。

GetDefaultPatchBaseline

操作 : `ssm:GetDefaultPatchBaseline`

查看默认补丁基准的信息所必需的。

GetDeployablePatchSnapshotForInstance

操作 : `ssm:GetDeployablePatchSnapshotForInstance`

查看实例使用的补丁基准的当前快照所必需的。主要由 AWS-RunPatchBaseline Systems Manager 文档使用。

GetDocument

操作 : `ssm:GetDocument`

查看指定 Systems Manager 文档内容所必需的。

GetInventory

操作 : `ssm:GetInventory`

查看清单项目信息所必需的。

GetInventorySchema

操作 : `ssm:GetInventorySchema`

查看账户清单类型名称或返回特定清单项目类型属性名称列表所必需的。

GetMaintenanceWindow

操作 : `ssm:GetMaintenanceWindow`

查看指定 Maintenance Window 的信息所必需的。

[GetMaintenanceWindowExecution](#)

操作 : `ssm:GetMaintenanceWindowExecution`

查看作为 Maintenance Window 执行的一部分运行的特定任务的信息所必需的。

[GetMaintenanceWindowExecutionTask](#)

操作 : `ssm:GetMaintenanceWindowExecutionTask`

查看作为 Maintenance Window 执行的一部分运行的特定任务的信息所必需的。

[GetMaintenanceWindowExecutionTaskInvocation](#)

操作 : `ssm:GetMaintenanceWindowExecutionTaskInvocation`

检索任务调用 (在特定目标上运行的特定任务) 所必需的。

[GetMaintenanceWindowTask](#)

操作 : `ssm:GetMaintenanceWindowTask`

列出 Maintenance Window 中的任务所必需的。

[GetParameter](#)

操作 : `ssm:GetParameter`

查看指定参数信息 (包括参数名称、类型和值) 所必需的。

[GetParameterHistory](#)

操作 : `ssm:GetParameterHistory`

查看指定参数历史信息所必需的。除了参数名称、类型和值以外，还返回参数描述、查询键 ID、上次修改日期和上次修改此参数的 AWS 用户的 ARN。

[GetParameters](#)

操作 : `ssm:GetParameters`

查看参数信息所必需的。

[GetParametersByPath](#)

操作 : `ssm:GetParametersByPath`

查看分层结构中参数的信息所必需的。

[GetPatchBaseline](#)

操作 : `ssm:GetPatchBaseline`

查看补丁基准的信息所必需的。

[GetPatchBaselineForPatchGroup](#)

操作 : `ssm:GetPatchBaselineForPatchGroup`

查看应该用于指定补丁组的补丁基准的信息所必需的。

[ListAssociations](#)

操作 : `ssm:ListAssociations`

查看指定 Systems Manager 文档或实例的关联所必需的。

ListAssociationVersions

操作 : `ssm:ListAssociationVersions`

检索特定关联 ID 的关联的所有版本所必需的。

ListCommandInvocations

操作 : `ssm:ListCommandInvocations`

查看调用列表或发送到特定实例的命令副本所必需的。

ListCommands

操作 : `ssm:ListCommands`

查看 AWS 账户用户请求的命令列表所必需的。

ListComplianceItems

操作 : `ssm:ListComplianceItems`

为特定资源 ID 检索不同资源类型的合规性状态列表所必需的。

ListComplianceSummaries

操作 : `ssm:ListComplianceSummaries`

检索合规性类型的合规资源和不合规资源的摘要计数所必需的。

ListDocuments

操作 : `ssm:ListDocuments`

查看 Systems Manager 文档列表所必需的。

ListDocumentVersions

操作 : `ssm:ListDocumentVersions`

查看文档版本信息所必需的。

ListInventoryEntries

操作 : `ssm:ListInventoryEntries`

查看实例上清单项目的信息所必需的。

ListResourceComplianceSummaries

操作 : `ssm:ListResourceComplianceSummaries`

检索资源级摘要计数 (包括合规和不合规状态信息) 所必需的。

ListResourceDataSync

操作 : `ssm:ListResourceDataSync`

查看资源数据同步配置的信息 (包括上次尝试启动同步的时间、上次同步状态以及上次成功完成同步的时间) 所必需的。

ListTagsForResource

操作 : `ssm:ListTagsForResource`

查看分配给指定资源的标签列表所必需的。

ModifyDocumentPermission

操作 : `ssm:ModifyDocumentPermission`

公开或私密共享 Systems Manager 文档所必需的。

[PutComplianceItems](#)

操作 : `ssm:PutComplianceItems`

在指定资源上注册合规性类型和其他合规性详细信息所必需的。

[PutInventory](#)

操作 : `ssm:PutInventory`

在一个或多个实例上添加或更新自定义清单项目所必需的。

[PutParameter](#)

操作 : `ssm:PutParameter`

向系统添加一个或多个参数所必需的。

[RegisterDefaultPatchBaseline](#)

操作 : `ssm:RegisterDefaultPatchBaseline`

定义默认补丁基准所必需的。

[RegisterPatchBaselineForPatchGroup](#)

操作 : `ssm:RegisterPatchBaselineForPatchGroup`

为补丁组注册补丁基准所必需的。

[RegisterTargetWithMaintenanceWindow](#)

操作 : `ssm:RegisterTargetWithMaintenanceWindow`

向 Maintenance Window 注册目标所必需的。

[RegisterTaskWithMaintenanceWindow](#)

操作 : `ssm:RegisterTaskWithMaintenanceWindow`

向 Maintenance Window 注册任务所必需的。

[RemoveTagsFromResource](#)

操作 : `ssm:RemoveTagsFromResource`

从指定资源中删除标签所必需的。

[SendAutomationSignal](#)

操作 : `ssm:SendAutomationSignal`

向自动化执行发送信号以更改执行的当前行为或状态所必需的。

[SendCommand](#)

操作 : `ssm:SendCommand`

在一个或多个托管实例上运行命令所必需的。

[StartAutomationExecution](#)

操作 : `ssm:StartAutomationExecution`

开始运行自动化文档所必需的。

[StopAutomationExecution](#)

操作 : `ssm:StopAutomationExecution`

开始运行自动化文档所必需的。

UpdateAssociation

操作 : `ssm:UpdateAssociation`

更新关联所必需的。只能更新文档版本、计划、参数和关联的 Amazon S3 输出。

UpdateAssociationStatus

操作 : `ssm:UpdateAssociationStatus`

更新与指定实例关联的 Systems Manager 文档的状态所必需的。

UpdateDocument

操作 : `ssm:UpdateDocument`

更新文档内容、版本或名称所必需的。

UpdateDocumentDefaultVersion

操作 : `ssm:UpdateDocumentDefaultVersion`

设置文档默认版本所必需的。

UpdateMaintenanceWindow

操作 : `ssm:UpdateMaintenanceWindow`

更新 Maintenance Window 中的一个或多个参数所必需的。

UpdateMaintenanceWindowTarget

操作 : `ssm:UpdateMaintenanceWindowTarget`

修改现有 Maintenance Window 的目标所必需的。

UpdateMaintenanceWindowTask

操作 : `ssm:UpdateMaintenanceWindowTask`

修改分配给 Maintenance Window 的任务所必需的。

UpdateManagedInstanceRole

操作 : `ssm:UpdateManagedInstanceRole`

向托管实例分配 Amazon Identity and Access Management (IAM) 角色或更改分配的 IAM 角色所必需的。

UpdatePatchBaseline

操作 : `ssm:UpdatePatchBaseline`

更新现有补丁基准中的一个或多个字段所必需的。

AWS Systems Manager 参考

以下信息和主题可以帮助您更好地实施 Systems Manager 解决方案。

委托人

在 AWS Identity and Access Management (IAM) 中，您可使用 Principal 策略元素授予或拒绝授予针对资源的服务访问权限。Systems Manager 的 Principal 策略元素的值是 `ssm.amazonaws.com`。

支持的区域和终端节点

请参阅 Amazon Web Services General Reference 中的 [AWS Systems Manager](#)。

服务限制

请参阅 Amazon Web Services General Reference 中的 [AWS Systems Manager](#)。

API 参考指南

请参阅 [AWS Systems Manager API Reference](#)。

CLI 命令参考

请参阅 [AWS Systems Manager section of the AWS CLI Command Reference](#)。

AWS Tools for PowerShell Cmdlet 参考

请参阅 [AWS Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference](#)。

GitHub 上的 SSM 代理 存储库

请参阅 [aws/amazon-ssm-agent](#)。

提问

[AWS Systems Manager 开发人员论坛](#)

AWS 新闻博客

管理工具

更多参考主题

- [参考：Systems Manager 的 Cron 和 Rate 表达式 \(p. 418\)](#)
- [参考：Ec2messages 和其他 API 调用 \(p. 422\)](#)

参考：Systems Manager 的 Cron 和 Rate 表达式

当您创建 AWS Systems Manager Maintenance Window 或 状态管理器 关联时，指定一个有关该时段或关联应何时运行的计划。您能够以基于时间的条目（也称为 cron 表达式）或基于频率的条目（也称为 rate 表达式）的形式指定计划。

如果您使用 Amazon EC2 控制台创建 Maintenance Window 或 关联，则可以使用用户界面中的工具来创建您的计划。如果您要以编程方式或从命令行使用 AWS CLI 等来创建 Maintenance Window 或 关联，则必须以正确格式指定带有 cron 或 rate 表达式的计划参数。

Note

要通过 AWS CLI 创建 Maintenance Window，请使用带有 cron 或 rate 表达式的 `--schedule` 参数。
要通过 AWS CLI 创建 状态管理器 关联，请使用带有 cron 或 rate 表达式的 `--scheduleExpression` 参数。

以下一些示例显示带有 cron 和 rate 表达式的 schedule 参数：

Cron 示例：此 cron 表达式在每个星期二下午 4 点 (16:00) 运行关联。

```
--scheduleExpression "cron(0 16 ? * TUE *)"
```

Rate 示例：此 rate 表达式每隔一天运行 Maintenance Window 或关联。

```
--schedule "rate(2 days)"
```

Important

与 Maintenance Window 相比，状态管理器 关联已限制 cron 和 rate 表达式的选项。在您为关联创建这些表达式之前，请查看以下部分中的限制。

如果您不熟悉 cron 和 rate 表达式，则建议您阅读[有关 Cron 和 Rate 表达式的一般信息 \(p. 420\)](#)。

主题

- [适用于关联的 Cron 和 Rate 表达式 \(p. 419\)](#)
- [适用于 Maintenance Window 的 Cron 和 Rate 表达式 \(p. 420\)](#)
- [有关 Cron 和 Rate 表达式的一般信息 \(p. 420\)](#)

适用于关联的 Cron 和 Rate 表达式

此部分包括适用于 状态管理器 关联的 cron 和 rate 表达式的示例。在您创建这些表达式之前，请注意以下限制。

- 关联仅支持以下 cron 表达式：每 1/2、1、2、4、8 或 12 个小时；每天或每周的特定时间。
- 关联仅支持以下 rate 表达式：30 分钟的时间间隔或大于和小于 31 天。

以下是一些适用于关联的 cron 示例。

适用于关联的 Cron 示例

示例	详细信息
cron(0/30 *** ? *)	每 30 分钟
cron(0 0/1 ** ? *)	每小时
cron(0 0/2 ** ? *)	每 2 小时
cron(0 0/4 ** ? *)	每 4 小时
cron(0 0/8 ** ? *)	每 8 小时
cron(0 0/12 ** ? *)	每 12 小时
cron(15 13 ? ** *)	每天下午 1:15
cron(15 13 ? * MON *)	每星期一下午 1:15

以下是一些适用于关联的 rate 示例。

适用于关联的 Rate 示例

示例	详细信息
rate(30 minutes)	每 30 分钟
rate(1 hour)	每小时
rate(5 hours)	每 5 小时
rate(15 days)	每 15 天

适用于Maintenance Window的 Cron 和 Rate 表达式

与 状态管理器 关联不同，Maintenance Window 支持所有 cron 和 rate 表达式。以下是一些适用于 Maintenance Window 的 cron 示例。

适用于Maintenance Window的 Cron 示例

示例	详细信息
cron(0 2 ? * THU#3 *)	每月第三个星期四凌晨 2:00
cron(15 10 ? * * *)	每天上午 10:15
cron(15 10 ? * MON-FRI *)	星期一到星期五每天上午 10:15
cron(0 2 L * ? *)	每月最后一天凌晨 2:00
cron(15 10 ? * 6L *)	每月最后一个星期五上午 10:15

以下是一些适用于Maintenance Window的 rate 示例。

适用于Maintenance Window的 Rate 示例

示例	详细信息
rate(30 minutes)	每 30 分钟
rate(1 hour)	每小时
rate(5 hours)	每 5 小时
rate(25 days)	每 25 天

有关 Cron 和 Rate 表达式的一般信息

适用于 Systems Manager 的 Cron 表达式有六个必需字段。这些字段用空格分隔。

分钟	小时	日期	月	星期几	年代	意义
0	10	*	*	?	*	每天上午的 10:00 (UTC) 运行

分钟	小时	日期	月	星期几	年代	意义
15	12	*	*	?	*	每天下午 12:15 (UTC) 运行
0	18	?	*	MON-FRI	*	每星期一到星期五下午 6:00 (UTC) 运行
0	8	1	*	?	*	每月第 1 天上午 8:00 (UTC) 运行
0/15	*	*	*	?	*	每 15 分钟运行一次
0/10	*	?	*	MON-FRI	*	从星期一到星期五，每 10 分钟运行一次
0/5	8-17	?	*	MON-FRI	*	在每星期一到星期五的上午 8:00 到下午 5:55 (UTC) 之间，每 5 分钟运行一次

下表显示了必需 cron 条目支持的值：

字段	值	通配符
分钟	0-59	, - * /
小时	0-23	, - * /
日期	1-31	, - * ? / L W
月	1-12 或 JAN-DEC	, - * /
星期几	1-7 或 SUN-SAT	, - * ? / L
年代	1970-2199	, - * /

Note

您无法在同一 cron 表达式中的“日期”和“星期几”字段中指定值。如果您在其中一个字段中指定了值，则必须在另一个字段中使用？(问号)。

通配符

Cron 表达式支持下列通配符：

- , (逗号) 通配符包含其他值。在“月份”字段中，JAN、FEB 和 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 将包含指定月份的 1 - 15 日。
- * (星号) 通配符包含该字段中的所有值。在“小时”字段中，* 将包含每个小时。
- / (正斜杠) 通配符用于指定增量。在“分钟”字段中，您可以输入 1/10 以指定从一个小时的第一分钟开始的每个第十分钟 (例如，第 11 分钟、第 21 分钟和第 31 分钟，依此类推)。
- 这些区域有：? (问号) 通配符用于指定一个或另一个。在“日期”字段中，您可以输入 7，如果您不介意 7 日是星期几，则可以在“星期几”字段中输入？。
- “日期”或“星期几”字段中的 L 通配符用于指定月或周的最后一天。
- “日期”字段中的 W 通配符用于指定工作日。在“日期”字段中，3W 用于指定最靠近当月的第三周的日。

Note

不支持产生的速率快于 5 分钟的 Cron 表达式。对指定星期几值和日期值的支持不完整。您当前必须在以下任一字段中使用“?”字符。

有关 cron 表达式的更多信息，请参阅 Wikipedia 网站上的 [CRON 表达式](#)。

Rate 表达式

Rate 表达式有以下两个必需字段。这些字段用空格分隔。

字段	值
值	正数
单位	分钟、小时或天

Note

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，而 rate(1 hour) 和 rate(5 hours) 有效。

参考：Ec2messages 和其他 API 调用

如果您监控 API 调用，将看到以下 API 调用。

- ec2messages:AcknowledgeMessage
- ec2messages>DeleteMessage
- ec2messages:FailMessage
- ec2messages:GetEndpoint
- ec2messages:GetMessages
- ec2messages:SendReply
- UpdateInstanceInformation
- ListInstanceAssociations
- DescribeInstanceProperties
- DescribeDocumentParameters

对 ec2messages:* API 的调用是对 ec2messages 终端节点的调用。Systems Manager 使用此终端节点从 SSM 代理调用云中的 Systems Manager 服务。需要此终端节点发送和接收命令。

UpdateInstanceInformation : SSM 代理每五分钟调用一次云中的 Systems Manager 服务以提供检测信号信息。需要此调用来保持代理的检测信号，以便服务了解代理按照预期工作。

ListInstanceAssociations : 代理调用此 API 以了解新的 Systems Manager 状态管理器 关联是否可用。状态管理器 需要此 API 才能正常运行。

DescribeInstanceProperties 和 **DescribeDocumentParameters** : Systems Manager 调用这些 API 在 Amazon EC2 控制台中呈现特定节点。**DescribeInstanceProperties** API 在左侧导航中显示托管实例节点。**DescribeDocumentParameters** API 在左侧导航中显示文档节点。

使用案例和最佳实践

此主题列出了 AWS Systems Manager 功能的常见使用案例和最佳实践。如果可用，本主题还包括指向相关博客文章和技术文档的链接。

Note

此处的每个部分标题都是一个指向技术文档中相应部分的有效链接。

[自动化 \(p. 100\)](#)

- 为基础设施创建 Automation 文档形式的自助服务运行手册。
- 使用 Automation 可对创建 AWS Marketplace 中的 AMI 或自定义 AMI、使用公共 SSM 文档或编写自己的工作流进行简化。
- 使用 [AWS-UpdateLinuxAmi](#) 和 [AWS-UpdateWindowsAmi](#) Automation 文档 (或创建自定义 Automation 文档) 可构建和维护 AMI。(p. 114)

[清单 \(p. 60\)](#)

- 将 Systems Manager Inventory 和 AWS Config 结合使用可随着时间的推移审核您的应用程序配置。

[Maintenance Window \(p. 246\)](#)

- 定义一个时间表以便在您的实例上执行具有潜在破坏性的操作，例如操作系统修补、驱动程序更新或软件安装。

[Parameter Store \(p. 361\)](#)

- 使用 Parameter Store 集中管理全局配置设置。
- 使用 Parameter Store 通过 AWS KMS 加密和管理密钥 (p. 382)。
- 将 Parameter Store 与 ECS 任务定义结合使用来存储密钥。

[Patch Manager \(p. 204\)](#)

- 使用补丁管理器大规模地推出补丁，并在您的实例中增加队列合规可见性。

[Run Command \(p. 171\)](#)

- 使用 EC2 Run Command 在不使用 SSH 访问的情况下大规模地管理实例。
- 使用 AWS CloudTrail 审核在 Run Command 上进行或代表其进行的所有 API 调用。
- 使用 Run Command 中的速率控制功能完成暂存的命令执行 (p. 185)。
- 借助 AWS Identity and Access Management (IAM) 策略，使用针对 Run Command (以及所有 Systems Manager 功能) 的精细访问权限 (p. 11)。

[状态管理器 \(p. 277\)](#)

- 使用预配置的 [AWS-UpdateSSMAgent](#) 文档，至少每月更新 SSM 代理一次。
- 使用适合 Windows 的 [EC2Config](#) 在启动时引导 EC2 实例
- (Windows) 将 PowerShell 或 DSC 模块上传到 Amazon S3，并使用 [AWS-InstallPowerShellModule](#)。

- 使用 Amazon EC2 标签为实例创建应用程序组。然后使用 `Targets` 参数而不是指定各个实例 ID 将实例设为目标。
- 使用 Systems Manager 自动修复 Amazon Inspector 生成的结果。
- 对所有 SSM 文档使用集中式配置存储库，在整个组织中共享文档 (p. 302)。

托管实例 (p. 289)

- Systems Manager 需要准确的时间引用以便执行其操作。如果实例的日期和时间设置不正确，它们可能与 API 请求的签名日期不匹配。在某些情况下，这会导致错误或功能不完整。例如，时间设置不正确的实例将不会包含在您的托管实例列表中。

有关在您的实例上设置时间的信息，请参阅以下主题之一：

- [为 Linux 实例设置时间](#)
- [为 Windows 实例设置时间](#)

文档历史记录

下表描述了自 AWS Systems Manager 上一次发布以来对文档所做的重要修改。如需对此文档更新的通知，您可以订阅 RSS 源。

- API 版本：2014-11-06
- 文档最新更新时间：2018 年 7 月 7 日

update-history-change	update-history-description	update-history-date
创建动态自动化工作流程 (p. 426)	默认情况下，您在自动化文档的 mainSteps 部分中定义的步骤（或操作）将按先后顺序执行。在一个操作完成后，mainSteps 部分中指定的下一个操作将开始。利用此版本，您现在可以创建执行条件分支的自动化工作流程。这意味着，您可以创建动态响应条件更改并跳转至指定步骤的自动化工作流程。有关信息，请参阅 创建动态自动化工作流程 。	July 11, 2018
SSM 代理现已通过 Snap 预安装到 Ubuntu Server 16.04 AMI 上 (p. 426)	从通过使用 20180627 标识的 Ubuntu Server 16.04 AMI 创建的实例开始，已使用 Snap 程序包预安装 SSM 代理。在通过以前的 AMI 创建的实例上，您应继续使用 deb 安装程序包。有关信息，请参阅 关于 64 位 Ubuntu Server 16.04 实例上的 SSM 代理安装 。	July 7, 2018
查看 SSM 代理所需的最低 S3 权限 (p. 426)	新的主题 SSM 代理的最低 S3 存储桶权限 提供了有关资源为执行 Systems Manager 操作可能需要访问的 Amazon Simple Storage Service (Amazon S3) 存储桶的信息。如果您想将实例配置文件或 VPC 终端节点的 Amazon S3 存储桶访问权限限制为使用 Systems Manager 所需的最小权限，则可在自定义策略中指定这些存储桶。	July 5, 2018
查看特定状态管理器关联 ID 的完整执行历史记录 (p. 426)	新的主题 查看关联历史记录 描述了如何查看特定关联 ID 的所有执行，然后查看一个或多个资源的执行详细信息。	July 2, 2018
Patch Manager 引入了对 Amazon Linux 2 的支持 (p. 426)	现在，您可以使用 Patch Manager 将补丁应用于 Amazon Linux 2 实例。有关 Patch Manager 操作系统支持的一般信息，请参阅 Patch Manager 支持的操作系统 。有关在定义补丁筛选条件时受支持的 Amazon Linux 2 键/值对的信息，请参阅 AWS	June 26, 2018

	Systems Manager API Reference 中的 PatchFilter 。	
向 Amazon CloudWatch Logs 发送命令输出 (p. 426)	为 Run Command 配置 Amazon CloudWatch Logs 这个新主题介绍了如何向 CloudWatch Logs 发送 Run Command 输出。	June 18, 2018
使用 AWS CloudFormation，快速创建或删除清单的资源数据同步 (p. 426)	您可以使用 AWS CloudFormation 创建或删除 Systems Manager 清单的资源数据同步。要使用 AWS CloudFormation，请将 AWS::SSM::ResourceDataSync 资源添加到您的 AWS CloudFormation 模板。相关详情，请参阅 AWS CloudFormation User Guide 中的 使用 AWS CloudFormation 模板 。您还可以手动创建清单的资源数据同步，如 配置清单的资源数据同步 中所述。	June 11, 2018
现在可通过 RSS 提供的 AWS Systems Manager 用户指南更新通知 (p. 426)	HTML 版本的 Systems Manager 用户指南现在支持更新的 RSS 源，此类更新显示在 Systems Manager 文档更新历史记录 页面。RSS 源包括 2018 年 6 月及此日期之后发布的更新。在此之前发布的更新仍会显示在 Systems Manager 文档更新历史记录页面。使用顶部菜单面板中的 RSS 按钮，订阅此源。	June 6, 2018
在脚本中指定一个退出代码，以重启托管实例 (p. 426)	新的主题 通过脚本重启托管实例 介绍了如何通过您在使用 Run Command 运行的脚本中指定退出代码，指示 Systems Manager 重启托管实例。	June 3, 2018
在删除自定义清单时通过 Amazon CloudWatch Events 创建事件	在 CloudWatch Events 中查看清单删除操作 这个新主题介绍了如何配置 Amazon CloudWatch Events 以在用户删除自定义清单时创建事件。	June 1, 2018

早期更新

下表描述了 2018 年 6 月之前每次发布 AWS Systems Manager User Guide 时进行的重要修改。

更改	描述	发行日期
创建 AWS 账户中所有托管实例的清单	您可以通过创建全局清单关联，轻松创建您的 AWS 账户中所有托管实例的清单。有关更多信息，请参阅 创建 AWS 账户中所有托管实例的清单 (p. 61) 。	2018 年 5 月 3 日

更改	描述	发行日期
	<p>Note</p> <p>全局清单关联在 SSM 代理版本 2.0.790.0 或更高版本中可用。有关如何在实例上更新 SSM 代理的信息，请参阅 示例：更新 SSM 代理 (p. 183)。</p>	
Ubuntu Server 18 上默认安装 SSM 代理	默认情况下，Ubuntu Server 18.04 LTS 64 位和 32 位 AMI 上安装了 SSM 代理。	2018 年 5 月 2 日
新主题	新的主题 发送使用文档版本参数的命令 (p. 184) 描述了如何使用 document-version 参数指定在运行命令时使用 SSM 文档的哪个版本。	2018 年 5 月 1 日
新主题	新主题 删除自定义清单 (p. 74) 介绍如何使用 AWS CLI 从 Amazon S3 中删除自定义清单数据。此主题还介绍如何使用 SchemaDeleteOption 通过禁用或删除自定义清单类型来管理自定义清单。此新功能使用 DeleteInventory API 操作。	2018 年 4 月 19 日
SSM 代理的 Amazon SNS 通知	您可以订阅 Amazon SNS 主题，以在新版本的 SSM 代理可用时收到通知。有关更多信息，请参阅 订阅 SSM 代理通知 (p. 29) 。	2018 年 9 月 4 日
CentOS 修补支持	Systems Manager 现在支持修补 CentOS 实例。有关支持的 CentOS 版本的信息，请参阅 Patch Manager 支持的操作系统 (p. 204) 。有关修补工作方式的更多信息，请参阅 Patch Manager 工作原理 (p. 205) 。	2018 年 3 月 29 日
新章节	为了提供 AWS Systems Manager User Guide 中参考信息的单个来源，引入了一个新的部分 AWS Systems Manager 参考 (p. 418) 。其他内容在推出后也将添加到此部分中。	2018 年 3 月 15 日
新主题	新主题 已批准补丁和已拒绝补丁列表的程序包名称格式 (p. 222) 详细说明了您可以在自定义补丁基准的已批准补丁和已拒绝补丁列表中输入的程序包名称格式。为 Patch Manager 支持的每种操作系统类型提供了示例格式。	2018 年 3 月 9 日
新主题	Systems Manager 现在与 Chef InSpec 集成。InSpec 是一个开源的运行时框架，您可以使用它在 GitHub 或 Amazon S3 上创建人工可读的配置文件。然后，您可以使用 Systems Manager 运行合规性扫描并查看合规和不合规的实例。有关更多信息，请参阅 将 Chef InSpec 配置文件与 Systems Manager Compliance 结合使用 (p. 55) 。	2018 年 3 月 7 日
新主题	新主题为 Systems Manager 使用服务相关角色 (p. 407) 介绍了如何将 AWS Identity and Access Management (IAM) 服务相关角色用于 Systems Manager。目前，仅在使用 Systems Manager Inventory 收集有关标签和资源组的元数据需要服务相关角色。	2018 年 2 月 27 日

更改	描述	发行日期
新增和更新的主题	<p>您现在可以使用 Patch Manager 来安装与实例上配置的默认源存储库不同的源存储库中的补丁。这可用于使用与安全性无关的更新、Ubuntu Server 的个人程序包存档 (PPA) 的内容、企业内部应用程序的更新等来修补实例。您在创建自定义补丁基准时指定备用补丁源存储库。有关更多信息，请参阅以下主题：</p> <ul style="list-style-type: none"> • 如何指定备用补丁源存储库 (Linux) (p. 208) • 创建默认补丁基准 (p. 225) • 创建对不同操作系统版本使用自定义存储库的补丁基准 (p. 235) <p>此外，您现在可以使用 Patch Manager 来修补 SUSE Linux Enterprise Server 实例。Patch Manager 支持修补 SLES 12.* 版本 (仅限 64 位)。有关更多信息，请参阅以下主题中特定于 SLES 的信息：</p> <ul style="list-style-type: none"> • 如何选择安全性补丁 (p. 205) • 如何安装补丁 (p. 209) • 补丁基准规则在 SUSE Linux Enterprise Server 上的工作原理 (p. 214) 	2018 年 2 月 6 日
新主题	<p>新的主题 在使用代理服务器的 Amazon Linux 实例上升级 Python 请求模块 (p. 28) 提供的说明可确保使用 Amazon Linux AMI 创建的实例已更新为 Python requests 模块的最新版本。此要求是为了确保与 Patch Manager 的兼容性。</p>	2018 年 1 月 12 日
新主题	<p>新的主题 介绍如何修补实例的 SSM 文档概述 (p. 215) 介绍了目前可用的七个 SSM 文档，有助于您使托管实例始终获得最新的安全相关更新补丁。</p>	2018 年 1 月 10 日
有关 Linux 支持的重要更新	<p>在各主题中更新了以下信息：</p> <ul style="list-style-type: none"> • 在 2017.09 及以后日期版本的 Amazon Linux 基本 AMI 上，默认情况下会安装 SSM 代理。 • 对于其他版本的 Linux (包括经 Amazon ECS 优化的 AMI 等非基本映像)，您必须手动安装 SSM 代理。 	2018 年 1 月 9 日
新主题	<p>新主题 关于 SSM 文档 AWS-RunPatchBaseline (p. 217) 详细介绍如何在 Windows 和 Linux 系统上使用此 SSM 文档。此外，还介绍了 AWS-RunPatchBaseline 文档中的两个可用参数 Operation 和 Snapshot ID。</p>	2018 年 1 月 5 日
新主题	<p>新增一节 Patch Manager 工作原理 (p. 205)，其中介绍一些技术细节，说明 Patch Manager 如何确定安装哪些安全补丁，以及它如何在每个支持的操作系统上安装这些补丁。此外，还介绍补丁基准规则在不同的 Linux 操作系统发行版上的工作原理</p>	2018 年 1 月 2 日
重新命名并改动了 Systems Manager 自动化操作参考的位置	<p>根据客户反馈，“自动化操作参考”现更名为“Systems Manager 自动化文档参考”。此外，我们还将此参考移到了“共享资源”>“文档”节点中，使其更靠近 SSM 文档插件参考 (p. 312)。有关更多信息，请参阅 Systems Manager 自动化文档参考 (p. 334)。</p>	2017 年 12 月 20 日

更改	描述	发行日期
新增监控章节和内容	新增一章 使用 AWS Systems Manager 监控实例 (p. 389) ，介绍如何将指标和日志数据发送到 Amazon CloudWatch Logs。新增主题 将日志发送到 CloudWatch Logs (CloudWatch 代理) (p. 390) ，介绍如何将实例 (仅限 64 位 Windows Server 实例) 上的监控任务从 SSM 代理迁移到 CloudWatch 代理。	2017 年 12 月 14 日
新增章节	新增一章 AWS Systems Manager 的身份验证和访问控制 (p. 398) ，全面介绍如何使用 AWS Identity and Access Management (IAM) 和 AWS Systems Manager 通过使用凭证确保资源访问。这些凭证提供访问 AWS 资源所需的权限，如访问存储在 Amazon S3 存储桶中的数据、向 Amazon EC2 实例发送命令和读取 Amazon EC2 实例上的标签。	2017 年 12 月 11 日
左侧导航窗格的改动	本用户指南左侧导航窗格中的标题更改为与新 AWS Systems Manager 控制台 中的标题一致。	2017 年 12 月 8 日
因 re:Invent 2017 而产生的多次更改	<ul style="list-style-type: none"> 正式发布的 AWS Systems Manager: AWS Systems Manager (以前称作 Amazon EC2 Systems Manager) 是一个统一接口，您可通过该接口轻松地将操作数据集中到一起，并跨 AWS 资源自动完成任务。您可以在此访问新 AWS Systems Manager 控制台。有关更多信息，请参阅 什么是 AWS Systems Manager ? (p. 1)。 YAML 支持：您可以创建 YAML 格式的 SSM 文档。有关更多信息，请参阅 AWS Systems Manager 文档 (p. 289)。 	2017 年 11 月 29 日
使用 Run Command 为 EBS 卷拍摄启用 VSS 的快照	使用 Run Command，您可以为附加到 Amazon EC2 Windows 实例的所有 Amazon Elastic Block Store (Amazon EBS) 卷拍摄应用程序一致性快照。快照过程使用 Windows 卷影复制服务 (VSS) 对 VSS 感知应用程序进行映像级备份，包括这些应用程序和磁盘之间的待处理事务中的数据。此外，当需要备份所有已连接的卷时，无需关闭实例或断开与它们的连接。有关更多信息，请参阅 使用 Run Command 为 EBS 卷拍摄启用 VSS 的快照 (p. 47) 。	2017 年 11 月 20 日
通过使用 VPC 端点增强可用的 Systems Manager 安全性	您可以通过配置 Systems Manager 以使用接口 VPC 端点来改善托管实例 (包括混合环境中的托管实例) 的安全状况。接口终端节点由 PrivateLink 提供技术支持，该技术让您可以通过使用私有 IP 地址私下访问 Amazon EC2 和 Systems Manager API。PrivateLink 将托管实例、Systems Manager 和 EC2 之间的所有网络流量限制在 Amazon 网络内 (托管实例无法访问 Internet)。而且，您无需 Internet 网关、NAT 设备或虚拟专用网关。有关更多信息，请参阅 设置 Systems Manager 的 VPC 端点 (p. 13) 。	2017 年 11 月 7 日

更改	描述	发行日期
针对文件、服务、Windows 角色和 Windows 注册表的清单支持	<p>SSM 清单现在支持从您的托管实例中收集以下信息。</p> <ul style="list-style-type: none"> 文件：名称、大小、版本、安装日期、修改时间和上次访问时间等。 服务：名称、显示名称、状态、依赖服务、服务类型、启动类型等。 Windows 注册表：注册表项路径、值名称、值类型和值。 Windows 角色：名称、显示名称、路径、功能类型、安装状态等。 <p>在尝试为这些清单类型收集信息之前，请更新需要清点的实例上的 SSM 代理。通过运行最新版本的 SSM 代理，可确保您能够收集所有支持的清单类型的元数据。有关如何使用 状态管理器 更新 SSM 代理 的信息，请参阅演练：自动更新 SSM 代理 (CLI) (p. 286)。</p> <p>有关清单的更多信息，请参阅关于 Systems Manager Inventory (p. 60)。</p>	2017 年 11 月 6 日
对 Automation 文档的更新	<p>修复了 Systems Manager Automation 访问权限设置和配置相关信息中的几个问题。有关更多信息，请参阅设置 Automation (p. 107)。</p>	2017 年 10 月 31 日
GitHub 和 Amazon S3 集成	<p>运行远程脚本：Systems Manager 现在支持从私有或公有 GitHub 存储库以及从 Amazon S3 下载并运行脚本。使用 <code>AWS-RunRemoteScript</code> 预定义的 SSM 文档或 <code>aws:downloadContent</code> 自定义 SSM 文档中的插件，可以运行 Playbooks 以及 Python、Ruby 或 PowerShell 等等中的脚本。当使用 Systems Manager 在混合环境中自动完成 Amazon EC2 实例以及本地托管实例的配置和部署时，以上更改会进一步完善基础设施即代码。有关更多信息，请参阅合作伙伴和产品集成 (p. 38)。</p> <p>创建复合 SSM 文档：Systems Manager 现在支持从一个主要 SSM 文档运行一个或多个次要 SSM 文档。这些运行其他文档的主要文档称为复合文档。通过使用复合文档，可以为多个 AWS 账户创建并共享一组标准的次要 SSM 文档，从而完成引导启动防病毒软件或域连接实例等常见任务。您可以运行 Systems Manager、GitHub 或 Amazon S3 中存储的复合或次要文档。在创建复合文档后，可以使用 <code>AWS-RunDocument</code> 预定义 SSM 文档运行它。有关更多信息，请参阅创建复合文档 (p. 308) 和从远程位置运行文档 (p. 309)。</p> <p>SSM 文档插件引用：为了更便于访问，我们将 SSM 文档的 SSM 插件引用从 Systems Manager API 引用中移到用户指南中。有关更多信息，请参阅SSM 文档插件参考 (p. 312)。</p>	2017 年 10 月 26 日

更改	描述	发行日期
Parameter Store 中的参数版本支持	<p>在编辑参数时，Parameter Store 现在会自动将其版本号增加 1。您可以在 API 调用和 SSM 文档中指定参数名和特定版本号。如果不指定版本号，系统自动使用最新版本。</p> <p>参数版本针对意外更改参数的情况提供额外保护。您可以查看所有版本的值，并在必要时引用旧版本。还可以使用参数版本查看一段时间内更改参数的次数。有关更多信息，请参阅 使用参数版本 (p. 375)。</p>	2017 年 10 月 24 日
支持为 Systems Manager 文档添加标签	<p>您现在可以使用 AddTagsToResource API、AWS CLI 或 AWS Tools for Windows，使用键值对为 Systems Manager 文档添加标签。通过添加标签，可帮助您根据为特定资源分配的标签快速确定它们。这是对现有的对托管实例、Maintenance Window、Parameter Store 参数和补丁基线的标签支持的补充。新主题包括为 Systems Manager 文档添加标签 (p. 299)和使用标签控制对文档的访问权限 (p. 301)。</p>	2017 年 10 月 3 日
用于根据反馈修复错误或更新内容的各种文档更新	<ul style="list-style-type: none"> 使用适用于 Raspbian Linux 的信息更新了在混合环境中设置 AWS Systems Manager (p. 31)。 使用适用于 Windows 实例的新要求更新了设置 AWS Systems Manager (p. 6)。SSM 代理需要 Windows PowerShell 3.0 或更高版本才能在 Windows 实例上运行特定 SSM 文档(例如，AWS-ApplyPatchBaseline 文档)。验证您的 Windows 实例是否在 Windows 管理框架 3.0 或更高版本上运行。该框架包括 PowerShell。有关更多信息，请参阅 Windows 管理框架 3.0。 	2017 年 10 月 2 日
使用 EC2Rescue 自动化工作流程对无法访问的 Windows 实例进行故障排除	<p>EC2Rescue 可以帮助您诊断并解决有关 Amazon EC2 Windows Server 实例的问题。使用 AWSSupport-ExecuteEC2Rescue 文档将工具作为 Systems Manager 自动化工作流程运行。AWSSupport-ExecuteEC2Rescue 文档旨在执行 Systems Manager 操作、AWS CloudFormation 操作和 Lambda 函数的组合，从而将使用 EC2Rescue 通常所需的步骤自动化。有关更多信息，请参阅 在无法访问的实例上运行 EC2Rescue 工具 (p. 133)。</p>	2017 年 9 月 29 日
默认情况下在 Amazon Linux 上安装了 SSM 代理	<p>在 2017.09 及以后日期版本的 Amazon Linux AMI 上，默认情况下会安装 SSM 代理。您必须其他版本的 Linux 上手动安装 SSM 代理，如在Linux 实例上安装和配置 SSM 代理 (p. 17)中所述。</p>	2017 年 9 月 27 日
Run Command 增强功能	<p>Run Command 包含以下增强功能。</p> <ul style="list-style-type: none"> 您可以通过创建一个 IAM 用户策略将命令执行限制为特定实例，该用户策略包含一个条件，规定用户只能对使用特定 Amazon EC2 标签标记的实例运行命令。有关更多信息，请参阅 基于实例标签限制 Run Command 访问 (p. 179)。 使用 Amazon EC2 标签将实例设为目标有更多选项。现在，您可以在发送命令时指定多个标签键和多个标签值。有关更多信息，请参阅 将命令发送到队列 (p. 185)。 	2017 年 9 月 12 日
Raspbian 上支持的 Systems Manager	<p>Systems Manager 现在可以在 Raspbian Jessie 和 Raspbian Stretch 设备上运行，包括 Raspberry Pi (32 位)。有关更多信息，请参阅 Raspbian (p. 25)。</p>	2017 年 9 月 7 日

更改	描述	发行日期
将 SSM 代理日志自动发送到 Amazon CloudWatch Logs	您现在可以在实例上进行简单的配置更改，以使 SSM 代理将日志文件发送到 CloudWatch。有关更多信息，请参阅 将日志发送到 CloudWatch Logs (SSM 代理) (p. 389)。	2017 年 9 月 7 日
加密资源数据同步	Systems Manager 资源数据同步可让您在数十或数百个托管实例上收集的清单数据聚合到一个中央 Amazon S3 存储桶。您现在可以使用 AWS Key Management Service 密钥加密资源数据同步。有关更多信息，请参阅 演练：使用资源数据同步聚合清单数据 (p. 86)。	2017 年 9 月 1 日
新的 状态管理器 演练	向 状态管理器 文档添加了两个新的演练： 演练：自动更新 SSM 代理 (CLI) (p. 286) 演练：在 EC2 Windows 实例上自动更新半虚拟化驱动程序 (控制台) (p. 287)	2017 年 8 月 31 日
Systems Manager 配置合规性	使用配置合规性扫描托管实例队列，了解补丁合规性和配置不一致性。您可以从多个 AWS 账户和区域中收集并聚合数据，然后深入了解不合规的特定资源。默认情况下，配置合规性将显示有关 Patch Manager 修补和 状态管理器 关联的合规性数据。您也可以根据 IT 或业务要求自定义服务并创建自己的合规性类型。有关更多信息，请参阅 AWS Systems Manager 配置合规性 (p. 92)。	2017 年 8 月 28 日
新的自动化操作： aws:executeAutomation	通过调用辅助自动化文件执行辅助自动化工作流程。借助此操作，您可以为最常见的工作流程创建自动化文档，并在自动化执行期间引用这些文档。因为无需跨类似文档复制步骤，此操作可以简化您的自动化文档。有关更多信息，请参阅 aws:executeAutomation (p. 351)。	2017 年 8 月 22 日
作为 CloudWatch 事件的目标的 Automation	您可通过指定 Automation 文档作为 Amazon CloudWatch 事件的目标来启动 Automation 工作流程。您可以按计划或者在特定 AWS 系统事件发生时启动工作流程。有关更多信息，请参阅 将 Automation 配置为 CloudWatch Events 目标 (可选) (p. 113)。	2017 年 8 月 21 日
状态管理器 关联版本控制和常规更新	您现在可以创建不同的 状态管理器 关联版本。每个关联具有 1000 个版本的限制。您还可为您的关联指定名称。此外，状态管理器 文档已更新，处理了过时信息和不一致之处。有关更多信息，请参阅 AWS Systems Manager 状态管理器 (p. 277)。	2017 年 8 月 21 日

更改	描述	发行日期
Maintenance Window 更改	<p>Maintenance Window 包括以下更改或增强：</p> <ul style="list-style-type: none"> 以前，Maintenance Window 只能通过使用 Run Command 执行任务。现在您可使用 Systems Manager Automation、AWS Lambda 和 AWS Step Functions 来执行任务。 您可编辑 Maintenance Window 的目标，指定目标名称、描述和所有者。 您可以编辑 Maintenance Window 中的任务，包括为 Run Command 和 Automation 任务指定新的 SSM 文档。 现在支持所有 Run Command 参数，包括 DocumentHash、DocumentHashType、TimeoutSeconds、Comment 和 NotificationConfig。 您现在可在尝试取消注册目标时使用 <code>safe</code> 标志。如果启用，系统将在任何任务引用目标时返回错误。 <p>有关更多信息，请参阅 AWS Systems Manager Maintenance Window (p. 246)。</p>	2017 年 8 月 16 日
新的 Automation 操作：aws:approve	<p>此新的 Automation 文档操作会临时暂停 Automation 执行，直至指定委托人批准或拒绝操作。在达到所需批准数后，Automation 执行将恢复。</p> <p>有关更多信息，请参阅 Systems Manager 自动化文档参考 (p. 334)。</p>	2017 年 8 月 10 日
不再需要 Automation 代入角色	<p>Automation 之前需要您指定服务角色 (或代入角色)，以便服务有权代表您执行操作。Automation 不再需要此角色，因为服务现在通过使用已调用执行的用户的上下文来操作。</p> <p>不过，以下情况仍需要您为 Automation 指定服务角色：</p> <ul style="list-style-type: none"> 当您想限制用户对资源的权限，但希望用户运行需要更高权限的自动化工作流程时。在此方案中，您可以创建具有更高权限的服务角色并允许用户运行此工作流程。 运行时长预计将超过 12 小时的操作需要一个服务角色。 <p>有关更多信息，请参阅 设置 Automation (p. 107)。</p>	2017 年 8 月 3 日
配置合规性	<p>使用 Amazon EC2 Systems Manager 配置合规性扫描托管实例队列，了解补丁合规性和配置不一致性。您可以从多个 AWS 账户和区域中收集并聚合数据，然后深入了解不合规的特定资源。有关更多信息，请参阅 AWS Systems Manager 配置合规性 (p. 92)。</p>	2017 年 8 月 8 日

更改	描述	发行日期
SSM 文档增强功能	<p>SSM 命令和策略文档现在提供跨平台支持。也就是说单个 SSM 文档可以处理 Windows 和 Linux 操作系统的插件。跨平台支持可以让您合并您管理的文档数量。使用 2.2 版或更高版本架构的 SSM 文档中提供跨平台支持。</p> <p>使用 2.0 版或更高版本架构的 SSM 命令文档现在包含相同类型的多个插件。例如，您可以创建多次调用 aws:runRunShellScript 插件的命令文档。</p> <p>有关 2.2 版架构变化的更多信息，请参阅 AWS Systems Manager 文档 (p. 289)。有关 SSM 插件的更多信息，请参阅 Systems Manager 插件。</p>	2017 年 12 月 7 日
Linux 修补	<p>Patch Manager 现在可以修补以下 Linux 发行版：</p> <p>64 位和 32 位系统</p> <ul style="list-style-type: none"> • Amazon Linux 2014.03、2014.09 或更高版本 • Ubuntu Server 16.04 LTS、14.04 LTS 或 12.04 LTS • Red Hat Enterprise Linux (RHEL) 6.5 或更高版本 <p>仅 64 位系统</p> <ul style="list-style-type: none"> • Amazon Linux 2015.03、2015.09 或更高版本 • Red Hat Enterprise Linux (RHEL) 7.x 或更高版本 <p>有关更多信息，请参阅 AWS Systems Manager Patch Manager (p. 204)。</p> <p>Note</p> <ul style="list-style-type: none"> • 要修补 Linux 实例，您的实例必须运行 SSM 代理 2.0.834.0 版或更高版本。有关更新此代理的信息，请参阅从控制台运行命令 (p. 182)中标题为示例：更新 SSM 代理的部分。 • AWS-ApplyPatchBaseline SSM 文档将替换为 AWS-RunPatchBaseline 文档。 	2017 年 6 月 7 日
资源数据同步	<p>您可以使用 Systems Manager 资源数据同步将从所有托管实例收集的清单数据存储到单个 Amazon S3 存储桶。收集新的清单数据后，资源数据同步自动更新集中式数据。所有清单数据存储在目标 Amazon S3 存储桶中后，您可以使用 Amazon Athena 和 Amazon QuickSight 等服务查询和分析聚合数据。有关更多信息，请参阅 配置清单的资源数据同步 (p. 71)。有关如何使用资源数据同步的示例，请参阅 演练：使用资源数据同步聚合清单数据 (p. 86)。</p>	2017 年 6 月 29 日

更改	描述	发行日期
Systems Manager 参数层次结构	<p>以平面列表的形式管理几十个或数百个 Systems Manager 参数十分耗时且容易出错。您可以使用参数层次结构来帮助组织和管理 Systems Manager 参数。一个层次结构是一个参数名称，包括您使用正斜杠定义的路径。下面是一个示例，它在名称中使用三个层次结构级别来标识以下内容：</p> <p>/环境/计算机类型/应用程序/数据</p> <div>/Dev/DBServer/MySQL/db-string13</div> <p>有关更多信息，请参阅 将参数组织成层次结构 (p. 366)。有关如何使用参数层次结构的示例，请参阅演练：使用层次结构管理参数 (AWS CLI) (p. 386)。</p>	2017 年 6 月 22 日
对 SUSE Linux Enterprise Server 的 SSM 代理支持	<p>您可以在 64 位 SUSE Linux Enterprise Server (SLES) 上安装 SSM 代理。有关更多信息，请参阅 在 Linux 实例上安装和配置 SSM 代理 (p. 17)。</p>	2017 年 6 月 14 日

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the AWS General Reference.