<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action &amp; Change</td>
<td>278</td>
</tr>
<tr>
<td>Application Management</td>
<td>5</td>
</tr>
<tr>
<td>Inventory</td>
<td>704</td>
</tr>
<tr>
<td>Maintenance Windows</td>
<td>620</td>
</tr>
<tr>
<td>Change Calendar</td>
<td>613</td>
</tr>
<tr>
<td>Automation</td>
<td>278</td>
</tr>
<tr>
<td>AppConfig</td>
<td>175</td>
</tr>
<tr>
<td>Parameter Store</td>
<td>198</td>
</tr>
<tr>
<td>Actions &amp; Change</td>
<td>278</td>
</tr>
<tr>
<td>Automation</td>
<td>279</td>
</tr>
<tr>
<td>Parameter types and examples</td>
<td>199</td>
</tr>
<tr>
<td>SecureString parameters</td>
<td>206</td>
</tr>
<tr>
<td>Native parameter support for Amazon Machine Image IDs</td>
<td>210</td>
</tr>
<tr>
<td>Public parameters</td>
<td>212</td>
</tr>
<tr>
<td>Getting started</td>
<td>224</td>
</tr>
<tr>
<td>Working with parameters</td>
<td>240</td>
</tr>
<tr>
<td>Parameter Store walkthroughs</td>
<td>266</td>
</tr>
<tr>
<td>CloudWatch dashboards</td>
<td>168</td>
</tr>
<tr>
<td>Trusted Advisor and PHD</td>
<td>168</td>
</tr>
<tr>
<td>Application Management</td>
<td>5</td>
</tr>
<tr>
<td>Resource Groups</td>
<td>170</td>
</tr>
<tr>
<td>Viewing operations data for AWS Resource Groups</td>
<td>172</td>
</tr>
<tr>
<td>AppConfig</td>
<td>175</td>
</tr>
<tr>
<td>How can AppConfig benefit my organization?</td>
<td>176</td>
</tr>
<tr>
<td>What types of targets are supported?</td>
<td>176</td>
</tr>
<tr>
<td>Is there a charge to use AppConfig?</td>
<td>176</td>
</tr>
<tr>
<td>Do I have to change my application to work with AppConfig?</td>
<td>176</td>
</tr>
<tr>
<td>How does AppConfig work?</td>
<td>176</td>
</tr>
<tr>
<td>What are the Service Quotas of AppConfig?</td>
<td>178</td>
</tr>
<tr>
<td>Getting started with AppConfig</td>
<td>178</td>
</tr>
<tr>
<td>Working with AppConfig</td>
<td>182</td>
</tr>
<tr>
<td>Parameter Store walkthroughs</td>
<td>266</td>
</tr>
<tr>
<td>MAintenance Windows</td>
<td>613</td>
</tr>
<tr>
<td>Who should use Change Calendar?</td>
<td>613</td>
</tr>
<tr>
<td>Benefits of Change Calendar</td>
<td>614</td>
</tr>
<tr>
<td>Getting started with Change Calendar</td>
<td>614</td>
</tr>
<tr>
<td>Working with Change Calendar</td>
<td>615</td>
</tr>
<tr>
<td>Add Change Calendar dependencies to Automation documents</td>
<td>619</td>
</tr>
<tr>
<td>MAintenance Windows</td>
<td>620</td>
</tr>
<tr>
<td>Controlling access</td>
<td>621</td>
</tr>
<tr>
<td>Working with maintenance windows (console)</td>
<td>634</td>
</tr>
<tr>
<td>Maintenance Windows tutorials (AWS CLI)</td>
<td>640</td>
</tr>
<tr>
<td>Maintenance window walkthroughs</td>
<td>685</td>
</tr>
<tr>
<td>Maintenance window scheduling and active period options</td>
<td>693</td>
</tr>
<tr>
<td>Instances &amp; Nodes</td>
<td>696</td>
</tr>
<tr>
<td>Compliance</td>
<td>696</td>
</tr>
<tr>
<td>Getting started with Configuration Compliance</td>
<td>697</td>
</tr>
<tr>
<td>Creating a Resource Data Sync for Configuration Compliance</td>
<td>697</td>
</tr>
<tr>
<td>Working with Configuration Compliance</td>
<td>699</td>
</tr>
<tr>
<td>Remediating compliance issues</td>
<td>702</td>
</tr>
<tr>
<td>Configuration Compliance walkthrough (AWS CLI)</td>
<td>703</td>
</tr>
<tr>
<td>Inventory</td>
<td>704</td>
</tr>
<tr>
<td>Learn more about inventory</td>
<td>707</td>
</tr>
<tr>
<td>Configuring Resource Data Sync for Inventory</td>
<td>712</td>
</tr>
</tbody>
</table>
What is AWS Systems Manager?

AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

A managed instance is a machine that has been configured for use with Systems Manager. Systems Manager also helps you configure and maintain your managed instances. Supported machine types include EC2 instances, on-premises servers, and virtual machines (VMs), including VMs in other cloud environments. Supported operating system types include Windows Server, multiple distributions of Linux, and Raspbian.

Using Systems Manager, you can associate AWS resources together by applying the same identifying resource tag to each of them. You can then view operational data for these resources as a resource group, to help monitor and troubleshoot.

For example, you can assign a resource tag of "Operation=North Region OS Patching" to all of the following resources:

- A group of EC2 instances
- A group of on-premises servers in your own facility
- A Systems Manager patch baseline that specifies which patches to apply to your managed instances
- An S3 bucket to store patching operation log output
- A Systems Manager maintenance window that specifies the schedule for the patching operation

After tagging the resources, you can view a consolidated dashboard in Systems Manager that reports the status of all the resources that are part of the patching operation in your North region. If a problem arises with any of these resources, you can take corrective action immediately.

Capabilities in Systems Manager

Systems Manager is comprised of individual capabilities (p. 3), which are grouped into five categories: Operations Management, Application Management, Actions & Change, Instances & Nodes, and Shared Resources.

This collection of capabilities is a powerful set of tools and features that you can use to perform many operational tasks. For example:

- Group AWS resources together by any purpose or activity you choose, such as application, environment, region, project, campaign, business unit, or software lifecycle.
- Centrally define the configuration options and policies for your managed instances.
- Centrally view, investigate, and resolve operational work items related to AWS resources.
- Automate or schedule a variety of maintenance and deployment tasks.
- Use and create runbook-style SSM documents that define the actions to perform on your managed instances.
- Run a command, with rate and error controls, that targets an entire fleet of managed instances.
• Securely connect to a managed instance with a single click, without having to open an inbound port or manage SSH keys.
• Separate your secrets and configuration data from your code by using parameters, with or without encryption, and then reference those parameters from a number of other AWS services.
• Perform automated inventory by collecting metadata about your Amazon EC2 and on-premises managed instances. Metadata can include information about applications, network configurations, and more.
• View consolidated inventory data from multiple AWS Regions and accounts that you manage.
• Quickly see which resources in your account are out of compliance and take corrective action from a centralized dashboard.
• View active summaries of metrics and alarms for your AWS resources.

Systems Manager simplifies resource and application management, shortens the time to detect and resolve operational problems, and helps you operate and manage your AWS infrastructure securely at scale.

Note
AWS Systems Manager was formerly known as Amazon Simple Systems Manager (SSM) and Amazon EC2 Systems Manager (SSM). For more information, see Systems Manager Service Name History (p. 2).

What is AWS Systems Manager? (Video)
View more AWS videos on the Amazon Web Services YouTube Channel.

Systems Manager Supported Regions
AWS Systems Manager is available in the AWS Regions listed in Systems Manager service endpoints in the Amazon Web Services General Reference. Before starting your Systems Manager configuration process, we recommend that you ensure the service is available in each of the AWS Regions you want to use it in.

For on-premises servers and VMs in your hybrid environment, we recommend that you choose the Region closest to your data center or computing environment.

Systems Manager Pricing
Some Systems Manager capabilities charge a fee. For more information, see AWS Systems Manager Pricing.

Systems Manager Service Name history
AWS Systems Manager (Systems Manager) was formerly known as "Amazon Simple Systems Manager (SSM)" and "Amazon EC2 Systems Manager (SSM)". The original abbreviated name of the service, "SSM", is still reflected in various AWS resources, including a few other service consoles. Some examples:

• Systems Manager Agent: SSM Agent
• Systems Manager parameters: SSM parameters
• Systems Manager service endpoints: ssm.us-east-2.amazonaws.com
• AWS CloudFormation resource types: AWS::SSM::Document
• AWS Config rule identifier: EC2_INSTANCE_MANAGED_BY_SSM
• AWS CLI commands: aws ssm describe-patch-baselines
• AWS Identity and Access Management (IAM) managed policy names: AmazonSSMReadOnlyAccess
• Systems Manager resource ARNs: arn:aws:ssm:us-east-2:111222333444:patchbaseline/pb-07d8884178EXAMPLE
Related Content

The following resources can help you work directly with Systems Manager.

- **AWS Blog & Podcast** – Read blog posts about Systems Manager in the AWS Management Tools Category, as well as other posts that are tagged with #Systems Manager.
- **Systems Manager Developer Forum** – Follow announcements, or post or answer a question in the AWS Systems Manager Forum.
- **AWS Systems Manager API Reference** – Provides descriptions, syntax, and usage examples for each of the Systems Manager actions and data types.
- **AWS Systems Manager section of the AWS CLI Command Reference** – Manage Systems Manager from a command line tool. Available to use on Windows, Mac, and Linux/UNIX systems.
- **AWS Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference** – Manage Systems Manager with the same PowerShell tools that you use to manage your Windows, Linux, or Mac environments.
- **Systems Manager service quotas** in the Amazon Web Services General Reference – Provides the default quotas for Systems Manager for an AWS account. Unless otherwise noted, each quota is Region-specific.

The following related resources can help you as you work with this service.

- **Classes & Workshops** – Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- **AWS Whitepapers** – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- **AWS Support Center** – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
- **AWS Support** – The primary web page for information about AWS Support, a one-on-one, fast-response support channel to help you build and run applications in the cloud.
- **Contact Us** – A central contact point for inquiries concerning AWS billing, account, events, abuse, and other issues.
- **AWS Site Terms** – Detailed information about our copyright and trademark; your account, license, and site access; and other topics.

Learn more about Systems Manager

- Systems Manager capabilities (p. 3)
- How Systems Manager works (p. 7)
- About SSM Agent (p. 10)
- Supported operating systems (p. 11)
- Accessing Systems Manager (p. 13)
- Systems Manager prerequisites (p. 14)

Systems Manager capabilities

Systems Manager capabilities are grouped into the following capability types:

Topics
Quick Setup

Quick Setup (p. 16) is a tool you can use to quickly configure required security roles and commonly used Systems Manager capabilities on your EC2 instances. These capabilities help you manage and monitor the health of your instances while providing the minimum required permissions to get started. Specifically, Quick Setup helps you configure the following components on the instances you choose or target by using tags:

- AWS Identity and Access Management (IAM) instance profile roles for Systems Manager.
- A scheduled, bi-weekly update of SSM Agent.
- A scheduled collection of inventory metadata every 30 minutes.
- A daily scan of your instances to identify missing patches.
- A one-time installation and configuration of the Amazon CloudWatch agent.
- A scheduled, monthly update of the CloudWatch agent.

Operations Management

Operations Management is a suite of capabilities that help you manage your AWS resources.

Explorer

Explorer (p. 118) is a customizable operations dashboard that reports information about your AWS resources. Explorer displays an aggregated view of operations data (OpsData) for your AWS accounts and across Regions. In Explorer, OpsData includes metadata about your EC2 instances, patch compliance details, and operational work items (OpsItems). Explorer provides context about how OpsItems are distributed across your business units or applications, how they trend over time, and how they vary by category. You can group and filter information in Explorer to focus on items that are relevant to you and that require action. When you identify high priority issues, you can use Systems Manager OpsCenter to run Automation runbooks and quickly resolve those issues.

OpsCenter

OpsCenter (p. 138) provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. OpsCenter is designed to reduce mean time to resolution for issues impacting AWS resources. This Systems Manager capability aggregates and standardizes OpsItems across services while providing contextual investigation data about each OpsItem, related OpsItems, and related resources. OpsCenter also provides Systems Manager Automation documents (runbooks) that you can use to quickly resolve issues. You can specify searchable, custom data for each OpsItem. You can also view automatically-generated summary reports about OpsItems by status and source.

CloudWatch Dashboards

Amazon CloudWatch Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources.
Trusted Advisor & Personal Health Dashboard (PHD)

Systems Manager hosts two online tools to help you provision your resources and monitor your account for health events. Trusted Advisor is an online tool that provides you real time guidance to help you provision your resources following AWS best practices. For more information, see Trusted Advisor.

The AWS Personal Health Dashboard provides information about AWS Health events that can affect your account. The information is presented in two ways: a dashboard that shows recent and upcoming events organized by category, and a full event log that shows all events from the past 90 days. For more information, see Getting Started with the AWS Personal Health Dashboard.

Application Management

Application Management is a suite of capabilities that help you manage your applications running in AWS.

Resource Groups

**AWS Resource Groups:** An AWS resource is an entity you can work with in AWS, such as Systems Manager SSM documents, patch baselines, maintenance windows, parameters, and managed instances; an Amazon Elastic Compute Cloud (EC2) instance; an Amazon Elastic Block Store (Amazon EBS) volume; a security group; or an Amazon Virtual Private Cloud (VPC). A resource group is a collection of AWS resources that are all in the same AWS Region, and that match criteria provided in a query. You build queries in the Resource Groups console, or pass them as arguments to Resource Groups commands in the AWS CLI. With Resource Groups, you can create a custom console that organizes and consolidates information based on criteria that you specify in tags. You can also use groups as the basis for viewing monitoring and configuration insights in AWS Systems Manager.

AWS AppConfig

AppConfig (p. 175) helps you create, manage, and quickly deploy application configurations. AppConfig supports controlled deployments to applications of any size. You can use AppConfig with applications hosted on EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices. To prevent errors when deploying application configurations, AppConfig includes validators. A validator provides a syntactic or semantic check to ensure that the configuration you want to deploy works as intended. During a configuration deployment, AppConfig monitors the application to ensure that the deployment is successful. If the system encounters an error or if the deployment triggers an alarm, AppConfig rolls back the change to minimize impact for your application users.

Parameter Store

Parameter Store (p. 198) provides secure, hierarchical storage for configuration data and secrets management. You can store data such as passwords, database strings, EC2 instance IDs and Amazon Machine Image (AMI) IDs, and license codes as parameter values. You can store values as plain text or encrypted data. You can then reference values by using the unique name you specified when you created the parameter.

Actions & Change

Systems Manager provides the following capabilities for taking action against or changing your AWS resources.

Automation

Use Systems Manager Automation (p. 278) to automate common maintenance and deployment tasks. You can use Automation to create and update Amazon Machine Images, apply driver and
agent updates, reset passwords on Windows Server instance, reset SSH keys on Linux instances, and apply OS patches or application updates.

Change Calendar

**Change Calendar (p. 613)** lets you set up date and time ranges when actions you specify (for example, in **Systems Manager Automation (p. 278)** documents) may or may not be performed in your AWS account. In Change Calendar, these ranges are called events. When you create a Change Calendar entry, you are creating a **Systems Manager document (p. 1031)** of the type **ChangeCalendar**. In Change Calendar, the document stores iCalendar 2.0 data in plaintext format. Events that you add to the Change Calendar entry become part of the document.

Maintenance Windows

Use **Maintenance Windows (p. 620)** to set up recurring schedules for managed instances to run administrative tasks like installing patches and updates without interrupting business-critical operations.

**Instances & Nodes**

Systems Manager provides capabilities for managing your EC2 instances, your on-premises servers and virtual machines (VMs) in your hybrid environment, and other types of AWS resources (nodes).

Compliance

Use **Systems Manager Configuration Compliance (p. 696)** to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant. By default, Configuration Compliance displays compliance data about Patch Manager patching and State Manager associations. You can also customize the service and create your own compliance types based on your IT or business requirements.

Inventory

**Inventory Manager (p. 704)** automates the process of collecting software inventory from managed instances. You can use Inventory Manager to gather metadata about applications, files, components, patches, and more on your managed instances.

Managed Instances

A **managed instance (p. 25)** is any EC2 instance or on-premises machine—a server or a virtual machine (VM)—in your hybrid environment that is configured for Systems Manager. To set up managed instances, you need to install SSM Agent on your machines (if not installed by default) and configure AWS Identity and Access Management (IAM) permissions. On-premises machines also require an activation code.

Hybrid Activations

To set up servers and VMs in your hybrid environment as managed instances, you need to create a managed-instance activation (p. 43). After you complete the activation, you receive an activation code and ID. This code/ID combination functions like an Amazon EC2 access ID and secret key to provide secure access to the Systems Manager service from your managed instances.

Session Manager

Use **Session Manager (p. 770)** to manage your EC2 instances through an interactive one-click browser-based shell or through the AWS CLI. Session Manager provides secure and auditable instance management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your EC2 instances.
Run Command

Use **Systems Manager Run Command** (p. 829) to remotely and securely manage the configuration of your managed instances at scale. Use Run Command to perform on-demand changes like updating applications or running Linux shell scripts and Windows PowerShell commands on a target set of dozens or hundreds of instances.

State Manager

Use **Systems Manager State Manager** (p. 864) to automate the process of keeping your managed instances in a defined state. You can use State Manager to ensure that your instances are bootstrapped with specific software at startup, joined to a Windows domain (Windows Server instances only), or patched with specific software updates.

Patch Manager

Use **Patch Manager** (p. 911) to automate the process of patching your managed instances with both security related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for Microsoft applications.) This capability enables you to scan instances for missing patches and apply missing patches individually or to large groups of instances by using EC2 instance tags. Patch Manager uses *patch baselines*, which can include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install security patches on a regular basis by scheduling patching to run as a Systems Manager maintenance window task. For Linux operating systems, you can define the repositories that should be used for patching operations as part of your patch baseline. This allows you to ensure that updates are installed only from trusted repositories regardless of what repositories are configured on the instance. For Linux, you also have the ability to update any package on the instance, not just those that are classified as operating system security updates. For Windows Server, you can also use Patch Manager to update supported Microsoft applications.

Distributor

Use **Distributor** (p. 999) to create and deploy packages to managed instances. Distributor lets you package your own software—or find AWS-provided agent software packages, such as **AmazonCloudWatchAgent**—to install on AWS Systems Manager managed instances. After you install a package for the first time, you can use Distributor to completely uninstall and reinstall a new package version, or perform an in-place update that adds new or changed files only. Distributor publishes resources, such as software packages, to AWS Systems Manager managed instances.

**Shared Resources**

Systems Manager uses the following shared resources for managing and configuring your AWS resources.

Documents

A **Systems Manager document** (p. 1031) (SSM document) defines the actions that Systems Manager performs. SSM document types include **Command** documents, which are used by State Manager and Run Command, and **Automation** documents, which are used by Systems Manager Automation. Systems Manager includes dozens of pre-configured documents that you can use by specifying parameters at runtime. Documents can be expressed in JSON or YAML, and include steps and parameters that you specify.

**How Systems Manager works**

Diagram 1 below shows a general example of the different processes that Systems Manager performs when executing an action like sending a command to your fleet of servers or performing an inventory
of the applications running on your on-premises servers. Each Systems Manager capability, for example Run Command and Maintenance Windows, uses a similar process of set up, execution, processing, and reporting.

1. **Configure Systems Manager**: Use the Systems Manager console, SDK, AWS CLI, or AWS Tools for Windows PowerShell to configure, schedule, automate, and run actions that you want to perform on your AWS resources.

2. **Verification and processing**: Systems Manager verifies the configurations, including permissions, and sends requests to the SSM Agent running on your instances or servers in your hybrid environment. SSM Agent performs the specified configuration changes.

3. **Reporting**: SSM Agent reports the status of the configuration changes and actions to Systems Manager in the AWS cloud. Systems Manager then sends the status to the user and various AWS services, if configured.

**Diagram 1: General Example of Systems Manager Process Flow**
AWS Systems Manager User Guide

How it works

AWS Cloud

AWS Systems Manager Console

AWS SDKs

AWS Toolkit for Windows PowerShell

AWS CLI

AWS Region 1

AWS CloudTrail

Amazon S3

Amazon CloudWatch

Built-in insights

VPC

Amazon EC2 Windows instances

VPC

Amazon EC2 Linux instances

AWS Systems Manager

Linux servers and VMs

Windows servers and VMs

On-premises hybrid environment
About SSM Agent

AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure these resources. The agent processes requests from the Systems Manager service in the AWS Cloud, and then runs them as specified in the request. SSM Agent then sends status and execution information back to the Systems Manager service by using the Amazon Message Delivery Service (service prefix: ec2messages).

SSM Agent must be installed on each instance you want to use with Systems Manager. SSM Agent is preinstalled, by default, on instances created from the following Amazon Machine Images (AMIs):

- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
- Windows Server 2016 and 2019
- Amazon Linux
- Amazon Linux 2
- Ubuntu Server 16.04
- Ubuntu Server 18.04
- Amazon ECS-Optimized

On other AMIs, and on on-premises servers and virtual machines for your hybrid environment, you must install the agent manually, as described in the table below.

**Important**
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

<table>
<thead>
<tr>
<th>Operating system type</th>
<th>SSM Agent installation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Windows</strong></td>
<td>Windows AMIs published before November 2016 use the EC2Config service to process requests and configure instances. Unless you have a specific reason for using the EC2Config service or an earlier version of SSM Agent to process Systems Manager requests, we recommend that you download and install the latest version of the SSM Agent to each of your EC2 instances and managed instances in your hybrid environment. For more information, see Installing and configuring SSM Agent on Windows Server instances (p. 65).</td>
</tr>
<tr>
<td><strong>Linux</strong></td>
<td>SSM Agent is installed by default on Amazon Linux, Amazon Linux 2, Ubuntu Server 16.04, and Ubuntu Server 18.04 LTS base EC2 AMIs. You must manually install SSM Agent on other versions of Amazon EC2 for Linux, including non-base images like Amazon ECS-Optimized AMIs. For more information, see Installing and configuring SSM Agent on EC2 instances for Linux (p. 69).</td>
</tr>
</tbody>
</table>
## Supported operating systems

Your EC2 instances, on-premises servers, and virtual machines (VMs) must be running one of the following operating systems in order to be used with AWS Systems Manager.

### Operating System Types

- Windows Server (p. 11)
- Linux (p. 11)
- Raspbian (p. 13)

### Windows Server

<table>
<thead>
<tr>
<th>Version</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2008 R2</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2012 and 2012 R2</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>2019</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

### Linux

#### Amazon Linux

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012.03 – 2018.03</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

Beginning with version 2015.03, Amazon Linux is released in Intel 64-bit (x86_64) versions only.
### Amazon Linux 2

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0 and all later versions</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

### Ubuntu Server

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.04 LTS and 14.04 LTS</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>16.04 LTS and 18.04 LTS</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

### Debian Server

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jessie (8)</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stretch (9)</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

### Red Hat Enterprise Linux (RHEL)

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.5</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>7.7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>7.8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>8.0</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>8.1</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>8.2</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Oracle Linux

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.7</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CentOS

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3 and later 6.x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>versions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.1 and later 7.x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>versions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SUSE Linux Enterprise Server (SLES)

<table>
<thead>
<tr>
<th>Versions</th>
<th>Intel 32-bit (x86)</th>
<th>Intel 64-bit (x86_64)</th>
<th>ARM 64-bit (arm64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 and later 12.x</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>versions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Raspbian

<table>
<thead>
<tr>
<th>Version</th>
<th>ARM 32-bit (arm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jessie</td>
<td>✓</td>
</tr>
<tr>
<td>Stretch</td>
<td>✓</td>
</tr>
</tbody>
</table>

Accessing Systems Manager

You can work with Systems Manager in any of the following ways:

**Systems Manager Console**

The AWS Systems Manager console is a browser-based interface to access and use Systems Manager.

**AWS Command Line Tools**

The AWS command line tools let you issue commands at your system's command line to perform Systems Manager and other AWS tasks, and is supported on Windows, Linux, and macOS. Using the CLI can be faster and more convenient than using the console. The command line tools also are useful if you want to build scripts that perform AWS tasks.

AWS provides two sets of command line tools: the AWS Command Line Interface (AWS CLI) and the AWS Tools for Windows PowerShell. For information about installing and using the AWS CLI, see the

Note
On your Windows Server instances, Windows PowerShell 3.0 or later is required to run certain SSM documents (for example, the legacy AWS-ApplyPatchBaseline document). Verify that your Windows Server instances are running Windows Management Framework 3.0 or later. The framework includes PowerShell.

AWS SDKs
AWS provides software development kits (SDKs) that consist of libraries and sample code for various programming languages and platforms (for example, Java, Python, Ruby, .NET, iOS and Android, and others). The SDKs provide a convenient way to create programmatic access to Systems Manager. For information about the AWS SDKs, including how to download and install them, see Tools for Amazon Web Services.

Systems Manager prerequisites

The prerequisites for using AWS Systems Manager to manage your EC2 instances, on-premises servers, and virtual machines (VMs) are covered step by step in the Setting Up chapters of this user guide:

- Setting up AWS Systems Manager (p. 25)
- Setting up AWS Systems Manager for hybrid environments (p. 43)

This topic provides an overview of these prerequisites.

To complete prerequisites for using Systems Manager

1. Create an AWS account and configure the required IAM roles.
2. Verify that Systems Manager is supported in the AWS Regions where you want to use the service.
3. Verify that your instances run a supported operating system.
4. For EC2 instances, create an IAM instance profile and attach it to your machines.
5. For on-premises servers and VMs, create an IAM service role for a hybrid environment.
6. Verify that you are allowing HTTPS (port 443) outbound traffic to the Systems Manager endpoints.
7. (Recommended) Create a VPC endpoint in Amazon Virtual Private Cloud to use with Systems Manager.
8. On on-premises servers, VMs, and EC2 instances created from AMIs that are not supplied by AWS, install a Transport Layer Security (TLS) certificate.
9. For on-premises servers and VMs, register the machines with Systems Manager through the managed instance activation process.
10. Install or verify installation of SSM Agent on each of your managed instances.

Integration with IAM and Amazon EC2

User access to Systems Manager, its capabilities, and its resources are controlled through policies that you use or create in AWS Identity and Access Management (IAM). If you plan to use computing resources provided by AWS, and not only on-premises servers and virtual machines (VMs), you also need to understand Amazon Elastic Compute Cloud (Amazon EC2) before you set up Systems Manager for your organization. Understanding how these services work is essential to successfully set up Systems Manager.

For more information about Amazon EC2, see the following:
AWS Systems Manager User Guide
Prerequisites

• Amazon Elastic Compute Cloud (Amazon EC2)
• Getting Started with Amazon EC2 Linux Instances
• Getting Started with Amazon EC2 Windows Instances
• What is Amazon EC2? (Linux)
• What is Amazon EC2? (Windows)

For more information about IAM, see the following:

• AWS Identity and Access Management (IAM)
• Getting Started with IAM
• What is IAM?
AWS Systems Manager Quick Setup

Use AWS Systems Manager Quick Setup to quickly configure required security roles and commonly used Systems Manager capabilities on your EC2 instances. These capabilities help you manage and monitor the health of your instances while providing the minimum required permissions to get started. Specifically, Quick Setup helps you configure the following components on the instances you choose or target by using tags:

- AWS Identity and Access Management (IAM) instance profile roles for Systems Manager.
- A scheduled, bi-weekly update of SSM Agent.
- A scheduled collection of Inventory metadata every 30 minutes.
- A daily scan of your instances to identify missing patches.
- A one-time installation and configuration of the Amazon CloudWatch agent.
- A scheduled, monthly update of the CloudWatch agent.

To access Quick Setup, choose Quick Setup in the navigation pane of the Systems Manager console. You can also access Quick Setup by choosing AWS Systems Manager at the top of the navigation pane, and then choosing Get Started with Systems Manager as shown.

**Note**
You can change Quick Setup configurations at any time. Before you do, we recommend that you learn how to change configurations by using the Quick Setup Results page. For more information, see Working with Quick Setup results (p. 21).

Permissions roles

By default, Systems Manager doesn't have permission to communicate with or perform actions on your instances. You must grant access by using an AWS Identity and Access Management (IAM) instance profile and an IAM service role (or assume role). An instance profile is a container that passes IAM role information to an EC2 instance at launch. A service role enables Systems Manager to run commands on your instances. For more information about instance profiles, see Using Instance Profiles in the IAM User Guide. For more information about service roles, see Creating a Role to Delegate Permissions to an AWS Service.

You can choose to have Quick Setup create and configure these roles for you by choosing **Use the default role**. If you select an existing role, then that role must include IAM policies with, at minimum,
the permissions described in this topic. If you select existing roles and they don't have these permissions, then Quick Setup may fail to configure one or more selected components, or those components may fail to run correctly.

**Note**
Quick Setup doesn't override instance profiles that already exist on your instances.

## Details about the default instance profile

In the **Instance profile role** section, if you choose **Use the default role**, then Quick Setup creates a new IAM instance profile that uses the **AmazonSSMManagedInstanceCore** policy and one additional policy. The **AmazonSSMManagedInstanceCore** policy enables Systems Manager to perform the following actions on your instances.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeAssociation",
        "ssm:GetDeployablePatchSnapshotForInstance",
        "ssm:GetDocument",
        "ssm:DescribeDocument",
        "ssm:GetManifest",
        "ssm:GetParameter",
        "ssm:GetParameters",
        "ssm:ListAssociations",
        "ssm:ListInstanceAssociations",
        "ssm:PutInventory",
        "ssm:PutComplianceItems",
        "ssm:PutConfigurePackageResult",
        "ssm:UpdateAssociationStatus",
        "ssm:UpdateInstanceAssociationStatus",
        "ssm:UpdateInstanceInformation"
      ],
      "Resource": "*
    },
    {
      "Effect": "Allow",
      "Action": [
        "ssmmessages:CreateControlChannel",
        "ssmmessages:CreateDataChannel",
        "ssmmessages:OpenControlChannel",
        "ssmmessages:OpenDataChannel"
      ],
      "Resource": "*
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2messages:AcknowledgeMessage",
        "ec2messages:DeleteMessage",
        "ec2messages:FailMessage",
        "ec2messages:GetEndpoint",
        "ec2messages:GetMessages",
        "ec2messages:GetReplicaSetStatus",
        "ec2messages:PutReplicaSetLabel",
        "ec2messages:PutReplicaSetState",
        "ec2messages:SendReply"
      ],
      "Resource": "*
    }
  ]
}
```
Details about the service role

In the Systems Manager service role section, if you choose Use the default role, then Quick Setup creates a new IAM service role that includes the following policies. The first policy enables Systems Manager to perform the following actions on your instances.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:ListInstanceProfilesForRole",
                "iam:PassRole",
                "ec2:DescribeIamInstanceProfileAssociations",
                "iam:GetInstanceProfile",
                "ec2:DisassociateIamInstanceProfile",
                "ec2:AssociateIamInstanceProfile",
                "iam:AddRoleToInstanceProfile"
            ],
            "Resource": "*"
        }
    ]
}
```

The following policy enables Systems Manager to perform the actions in the previous policy on your behalf. Systems Manager assumes your role to perform the actions.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ssm.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```
Note
Configuring an instance with an instance profile for Systems Manager does not give a user access to run commands or use Systems Manager capabilities on that instance. Your IAM user, group, or role must be configured with a separate permissions policy that enables you to perform actions on your instances by using Systems Manager. For more information, see Setting up AWS Systems Manager (p. 25).

Update Systems Manager (SSM) Agent

SSM Agent is Amazon software that processes requests from the Systems Manager service in the AWS Cloud, and then runs them on your instance as specified in the request. SSM Agent is preinstalled, by default, on the following Amazon Machine Images (AMIs):

- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
- Windows Server 2016 and 2019
- Amazon Linux
- Amazon Linux 2
- Ubuntu Server 16.04
- Ubuntu Server 18.04
- Amazon ECS-Optimized

If you enable this option, then Systems Manager automatically checks every two weeks for a new version of the agent. If there is a new version, then Systems Manager automatically updates the agent on your instance to the latest released version. We encourage you to choose this option to ensure that your instances are always running the most up-to-date version of SSM Agent. For more information about SSM Agent, including information about how to manually install the agent, see Working with SSM Agent (p. 63).

Collect inventory from your instances

AWS Systems Manager Inventory provides visibility into your computing environment. You can use Inventory to collect metadata from your managed instances. You can store this metadata in a central Amazon Simple Storage Service (Amazon S3) bucket. Then use built-in tools to query the data and quickly determine which instances are running the software and configurations required by your software policy, and which instances need to be updated. Quick Setup configures collection of the following types of metadata:

- **AWS components**: EC2 driver, agents, versions, and more.
- **Applications**: Application names, publishers, versions, and more.
- **Instance details**: System name, operating system (OS) name, OS version, last boot, DNS, domain, work group, OS architecture, and more.
- **Network configuration**: IP address, MAC address, DNS, gateway, subnet mask, and more.
- **Services**: Name, display name, status, dependent services, service type, start type, and more (Windows Server instances only).
- **Windows roles**: Name, display name, path, feature type, installed state, and more (Windows Server instances only).
- **Windows updates**: Hotfix ID, installed by, installed date, and more (Windows Server instances only).

You can configure Systems Manager Inventory to collect the following additional types of metadata from your instances. For more information, see AWS Systems Manager Inventory (p. 704).
- **Custom inventory**: Metadata that was assigned to a managed instance as described in [Working with custom inventory](p. 736).
- **Files**: Name, size, version, installed date, modification time, last accessed time, and more.
- **Tags**: Tags assigned to your instances.
- **Windows Registry**: Registry key path, value name, value type, and value.

**Important**

A **global** Inventory configuration is created from the one-click configuration option on the Inventory page and targets all instances in your AWS account. If you already have a global Inventory configuration enabled on your instances, then enabling Inventory collection in Quick Setup creates another schedule to collect Inventory data for the specified or targeted instances. There is no conflict. However, if you already have a non-global Inventory configuration enabled on some of your instances, then the configuration you created earlier will fail. Inventory collection does not support more than one specific configuration per instance. To resolve this issue, either deploy a global inventory configuration or delete the conflict configuration.

**Scan instances for missing patches daily**

If you enable this option in Quick Setup, then Systems Manager uses Patch Manager to scan your instances each day and generate a simple report in the **Compliance** page. The report shows how many instances are patch-compliant according to the **default patch baseline**. The report includes a list of each instance and its compliance status. You can navigate this list to see details about noncompliant instances. For more information about patching operations and patch baselines, see [AWS Systems Manager Patch Manager](p. 911). To view compliance information, see the Systems Manager Compliance page.

**Install and configure the CloudWatch agent**

Amazon CloudWatch provides data and actionable insights to monitor your applications, understand and respond to system-wide performance changes, optimize resource utilization, and get a unified view of operational health. The CloudWatch agent collects metrics and log files from your instances and consolidates this information so that you can quickly determine the health of your instances. For more information, see [Collecting metrics and logs from EC2 instances and on-premises servers with the CloudWatch Agent](). There may be added cost. For more information, see [Amazon CloudWatch pricing]().

**Update the CloudWatch agent once every four weeks**

If you enable this option, then Systems Manager automatically checks every four weeks for a new version of the CloudWatch agent. If there is a new version, then Systems Manager automatically updates the agent on your instance to the latest released version. We encourage you to choose this option to ensure that your instances are always running the most up-to-date version of the CloudWatch agent.

**Choosing targets for Quick Setup**

After you choose Quick Setup options, choose which instances you want to configure with those options in the **Targets** section. Quick Setup includes the following options for targeting instances.
• **Choose all instances in the current AWS account and Region:** Quick Setup locates and applies the configuration options to all instances in the current AWS account and Region.

• **Specify instance tags:** Quick Setup uses the tag key and (optional) tag value that you specify to locate instances.

• **Choose instances manually:** Quick Setup enables you to choose one or more instances from a list.

**Note**

You can't use Quick Setup to configure options across accounts or Regions.

### Working with Quick Setup results

Systems Manager displays the results of Quick Setup in a separate card for each option you selected.

For each option you selected, Systems Manager creates and immediately runs a State Manager association. The **Configuration status** field shows **Success** when the association successfully runs on all selected or targeted instances. If one association fails to run, **Configuration status** is **Failed**. If an association is processing, **Configuration status** is **Pending**. To update **Configuration status** for **Pending** items, refresh your browser.
Note
The Inventory collection option can take up to 10 minutes to complete, even if you only selected a few instances.

A Configuration status of Not configured, means you didn't choose the option in Quick Setup. If you see this status for Managed instances, it means that you didn't choose a role in the Role selection list. You can run Quick Setup again, as described in this topic, and choose a role.

To edit the association for a Quick Setup option, choose the Edit button in the option card. If you edit the association, don't choose a different SSM Document in the Edit Association page. If you choose a different SSM Document, the option becomes unavailable in Quick Setup. Change only the parameters and targets of the association. When you save your changes to the association, State Manager automatically runs the association.

Troubleshooting Quick Setup results

If a Quick Setup card shows Not configured, you might have missed a selection on the Quick Setup page. As a first step in troubleshooting this problem, choose the Edit all button at the top of the Quick Setup results page and review your selections. If you missed one or more, you can choose them and then choose Reset to configure those options.

If you still see a problem with one or more Quick Setup results cards, then use the following procedure to troubleshoot the issue.

To troubleshoot a failed Quick Setup configuration

1. In the Quick Setup results page, choose View Details in the card with a Configuration status of Failed.
2. In the Association ID page, choose the Execution history tab.
3. Under Execution ID, choose the association execution that failed.
4. The **Association execution targets** page lists all of the instances where the association ran. Choose the **Output** button for an execution that failed to run.

5. In the **Output** page, choose **Step - Output** to view the error message for that step in the command execution. Each step can display a different error message. Review the error messages for all steps to help troubleshoot the issue.

If viewing the step output doesn't help troubleshoot the problem, then you can recreate the association to see if the problem persists. To recreate the association, you must first delete all associations that already exist for the Quick Setup option. You can delete an association by using the **Delete** button in the Quick Setup results card. You can also delete the association by choosing **State Manager** in the navigation pane. After you delete the association, run Quick Setup again to see if the problem persists.

### Running Quick Setup again

You can run Quick Setup again by choosing the **Edit all** button on the Quick Setup results page. You can choose or clear options. If you clear an option, Systems Manager doesn't delete the association. For example, if you previously selected the **Collect inventory from your instances every 30 minutes** option, and you clear the option when you run Quick Setup again, the original association still exists. You must either delete the association from the Quick Setup results page or from the **State Manager** page.
Important
If you want to run Quick Setup on new instances, we recommend that you choose the new instances and also choose all of the instances on which you previously ran Quick Setup. By choosing all of the instances on which you previously ran Quick Setup, you synchronize the association executions for all of the instances. This synchronizes status and compliance reporting. We also recommend that you target instances by using tags. New instances with the specified tags are automatically added to Quick Setup associations. This means that they automatically display status in the Quick Setup results and in the Compliance page.
Setting up AWS Systems Manager

This section describes the tasks that account and system administrators perform to set up AWS Systems Manager for their organizations. After these steps are complete, users in the organization can use Systems Manager to configure and manage the Amazon Elastic Compute Cloud (EC2) instances in their account.

If you plan to use Systems Manager to manage and configure your own on-premises servers and virtual machines (VMs) in what is called a hybrid environment, follow the setup steps in Setting up AWS Systems Manager for hybrid environments (p. 43). If you plan to use both EC2 instances and your own computing resources in a hybrid environment, follow the steps here first. This section presents steps in the best order for configuring the roles, users, permissions, and initial resources to use in your Systems Manager operations.

If you already use other AWS services, you have completed some of these steps. However, other steps are specific to Systems Manager. Therefore, we recommend reviewing this entire section to ensure that you are ready to use all Systems Manager capabilities.

Note
You can use Systems Manager Quick Setup to quickly configure an AWS Identity and Access Management (IAM) instance profile on all instances in your AWS account. Quick Setup can also create an assume role, which enables Systems Manager to securely run commands on your instances on your behalf. By using Quick Setup, you can skip step 4 and 5 in this section. For more information, see AWS Systems Manager Quick Setup (p. 16).

Contents
- Step 1: Sign up for AWS (p. 25)
- Step 2: Create an Admin IAM user for AWS (p. 26)
- Step 3: Create non-Admin IAM users and groups for Systems Manager (p. 27)
- Step 4: Create an IAM instance profile for Systems Manager (p. 30)
- Step 5: Attach an IAM instance profile to an EC2 instance (p. 35)
- Step 6: (Optional) Create a Virtual Private Cloud endpoint (p. 37)
- Step 7: (Optional) Create Systems Manager service roles (p. 40)
- Step 8: (Optional) Set up integrations with other AWS services (p. 42)

Step 1: Sign up for AWS

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

Continue to Step 2: Create an Admin IAM user for AWS (p. 26).
Step 2: Create an Admin IAM user for AWS

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

In this procedure, you use the AWS account root user to create your first user in AWS Identity and Access Management (IAM). You add this IAM user to an Administrators group, to ensure that you have access to all services and their resources in your account. The next time that you access your AWS account, you should sign in with the credentials for this IAM user.

To create an IAM user with restricted permissions, see Step 3: Create non-Admin IAM users and groups for Systems Manager (p. 27).

To create an administrator user for yourself and add the user to an administrators group (console)

1. Use your AWS account email address and password to sign in as the AWS account root user to the IAM console at https://console.aws.amazon.com/iam/.
   
   Note
   We strongly recommend that you adhere to the best practice of using the Administrator IAM user below and securely lock away the root user credentials. Sign in as the root user only to perform a few account and service management tasks.

2. In the navigation pane, choose Users and then choose Add user.

3. For User name, enter Administrator.

4. Select the check box next to AWS Management Console access. Then select Custom password, and then enter your new password in the text box.

5. (Optional) By default, AWS requires the new user to create a new password when first signing in. You can clear the check box next to User must create a new password at next sign-in to allow the new user to reset their password after they sign in.

6. Choose Next: Permissions.

7. Under Set permissions, choose Add user to group.

8. Choose Create group.

9. In the Create group dialog box, for Group name enter Administrators.

10. Choose Filter policies, and then select AWS managed -job function to filter the table contents.

11. In the policy list, select the check box for AdministratorAccess. Then choose Create group.

   Note
   You must activate IAM user and role access to Billing before you can use the AdministratorAccess permissions to access the AWS Billing and Cost Management console. To do this, follow the instructions in step 1 of the tutorial about delegating access to the billing console.

12. Back in the list of groups, select the check box for your new group. Choose Refresh if necessary to see the group in the list.

13. Choose Next: Tags.

14. (Optional) Add metadata to the user by attaching tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM Entities in the IAM User Guide.

15. Choose Next: Review to see the list of group memberships to be added to the new user. When you are ready to proceed, choose Create user.
You can use this same process to create more groups and users and to give your users access to your AWS account resources. To learn about using policies that restrict user permissions to specific AWS resources, see Access Management and Example Policies.

Continue to Step 3: Create non-Admin IAM users and groups for Systems Manager (p. 27).

Step 3: Create non-Admin IAM users and groups for Systems Manager

Users in the administrators group for an account have access to all AWS services and resources in that account. This section describes how to create users with permissions that are limited to AWS Systems Manager.

The following Systems Manager capabilities may have additional or alternative procedures for granting user access:

- **Session Manager** - See Control user session access to instances (p. 782).
- **Distributor** - See Control User Access to Packages (p. 1003).
- **Maintenance Windows** - See Controlling access to maintenance windows (p. 621) (see the instructions for assigning the IAM PassRole policy to an IAM user or group).

For more information about using IAM policies to control user access to Systems Manager capabilities and resources, see AWS Systems Manager identity-based policy examples (p. 1115).

For information about how to change permissions for an IAM user account, group, or role, see Changing Permissions for an IAM User in the IAM User Guide.

**Topics**

- Task 1: Create user groups (p. 27)
- Task 2: Create users and assign permissions (p. 28)

**Task 1: Create user groups**

You can create a user group for each policy and assign users to a group rather than attaching individual policies to each user.

You can create multiple user groups with different permission sets by omitting recommended or optional policies. You can also create custom IAM policies to grant any combination of permissions for a user. For example, you can grant a user group permission to use only the Session Manager capability in Systems Manager, as described in Control user session access to instances (p. 782).

For additional examples of custom IAM policies for Systems Manager, see Customer managed policy examples (p. 1117).

For comprehensive information about using IAM policies for Systems Manager access, see Identity and access management for AWS Systems Manager (p. 1105).

**To create a user group**

Use the following procedure to create a user group for your Systems Manager users. You can repeat this procedure to create additional user groups with different sets of permissions.
1. In the navigation pane of the IAM console, choose **Groups**, and then choose **Create New Group**.
2. On the **Set Group Name** page, enter a name for the group, such as **SSMUserGroup** or another name that you prefer.
3. Choose **Next Step**.
4. On the **Attach Policy** page, for **Filter**, enter **SSM**.
5. In the policy list, do the following:
   - If you want to provide users with permission to use Resource Groups and the Tag Editor, choose the **ResourceGroupsandTagEditorFullAccess** policy.
     
     AWS resource groups can be managed in the AWS Resource Groups service and in the Resource Groups capability in Systems Manager. It is optional to provide the users and user groups in your account access to this service and its Tag Editor, but we recommend it for more effective management operations.
     
     For more information, see the Resource Groups in AWS Systems Manager (p. 170).
   - To provide users in this group with full access to the Systems Manager console, select the box next to **AmazonSSMFullAccess**.
     
     -or-
   - If you want users in this group only to view Systems Manager data, and not create or update resources, select the box beside **AmazonSSMReadOnlyAccess**.
     
     • To provide users with access to the **Built-In Insights** and **Dashboard by CloudWatch** pages in the Systems Manager console, select the boxes next to these managed policies:
       - **AWSHealthFullAccess**
         
         This policy grants full access to the AWS Health APIs and Notifications and the Personal Health Dashboard. It also provides access to portions of the Built-In Insights Dashboard in the Systems Manager console.
       - **AWSConfigUserAccess**
         
         This policy provides read-only access to use AWS Config, including searching by tags on resources, and reading all tags. It also provides access to portions of the Built-In Insights Dashboard in the Systems Manager console.
       - **CloudWatchReadOnlyAccess**
         
         This policy provides read-only access to CloudWatch, which is needed to view information on the **Dashboard by CloudWatch** in the Systems Manager console.
   - Add any other policies that provide permissions you want to grant to this user group.
6. Choose **Next Step**.
7. On the **Review** page, verify that the correct policies are added to this group, and then choose **Create Group**.

Continue to Task 2: Create users and assign permissions (p. 28).

**Task 2: Create users and assign permissions**

Create IAM users for the individuals who require access to AWS Systems Manager, and add each user to the appropriate user group to ensure that they have the right level of permissions.

**Note**

If your organization has an existing identity system, you might want to create a single sign-on (SSO) option. SSO gives users access to the AWS Management Console for your account without requiring them to have an IAM user identity. SSO also eliminates the need for users to sign in
to your organization's site and to AWS separately. For more information, see Enabling Custom Identity Broker Access to the AWS Console.

Depending on whether the user accounts for this group were already created, use one of the following procedures:

**To create users and add permissions**

1. In the navigation pane of the IAM console, choose **Users**, and then choose **Add user**.
2. For **User name**, enter the name that the user will use to sign in to AWS Systems Manager.
3. To allow the user access to the AWS API, AWS CLI, AWS SDK, and other development tools, select the check box next to **Programmatic access**.

   This creates an access key for the new user. You can view or download the access keys when you get to the **Final page**.
4. To allow the user access to the AWS Management Console, select the check box next to **AWS Management Console access**.

   The AWS Management Console provides a web interface where you can manage your compute, storage, and other cloud resources. Within the AWS Management Console, individual services have their own console. For example, you can manage your compute resources using the Amazon EC2 console and storage through the Amazon S3 console.

   If you choose **Custom password**, enter an initial password for the user. You can optionally select **Require password reset** to force the user to create a new password the next time the user signs in.
5. Choose **Next: Permissions**.
6. On the **Set permissions for user** page, choose **Add user to group**.
7. In the group list, choose the user group to add the user to, and then choose **Next: Tags**.
8. (Optional) Add one or more tag key-value pairs to organize, track, or control access for this user, and then choose **Next: Review** to see the list of group memberships that the new user is joining.
9. Choose **Create user**.
10. To view the users' access keys (access key IDs and secret access keys), choose **Show** next to each password and access key that you want to see. To save the access keys, choose **Download .csv** and then save the file to a safe location.

   **Important**
   This is your only opportunity to view or download the secret access keys, and you must provide this information to your users before they can use the AWS API or AWS CLI. Save the user's new access key ID and secret access key in a safe and secure place. You will not have access to the secret keys again after this step.
11. Provide each user with his or her credentials. On the final page you can choose **Send email** next to each user. Your local mail client opens with a draft that you can customize and send. The email template includes the following details for each user:

   - User name
   - URL of the account sign-in page. Use the following example, substituting the correct account ID number or account alias:

     https://AWS-account-ID or alias.signin.aws.amazon.com/console

   For more information, see How IAM Users Sign In to AWS in the IAM User Guide.

   **Important**
   The user's password is not included in the generated email. You must provide them to the user in a way that complies with your organization's security guidelines.
To add permissions for an existing user

1. In the IAM console navigation pane, choose Users.
2. Choose the name of the user to add to a group, and then choose Add permission.
3. For Add user to group, select the box next to the group to add the user to, such as SSMUserGroup, or the name of a different user group that you created.
4. Add any other available permission policies to assign to the user.
5. Choose Next: Review to see the list of group memberships that will be added to the new user.
6. Choose Add permissions.

Continue to Step 4: Create an IAM instance profile for Systems Manager (p. 30).

Step 4: Create an IAM instance profile for Systems Manager

By default, AWS Systems Manager doesn't have permission to perform actions on your instances. You must grant access by using an AWS Identity and Access Management (IAM) instance profile. An instance profile is a container that passes IAM role information to an Amazon Elastic Compute Cloud (Amazon EC2) instance at launch. You can create an instance profile for Systems Manager by attaching one or more IAM policies that define the necessary permissions to a new role or to a role you already created.

Note
You can use Systems Manager Quick Setup to quickly configure an instance profile on all instances in your AWS account. Quick Setup can also create an assume role, which enables Systems Manager to securely run commands on your instances on your behalf. By using Quick Setup, you can skip this step (Step 4) and Step 5. For more information, see AWS Systems Manager Quick Setup (p. 16).

Note the following details about creating an IAM instance profile:

- If you are configuring servers or virtual machines (VMs) in a hybrid environment for Systems Manager, you don't need to create an instance profile for them. Instead, you must configure your servers and VMs to use an IAM service role. For more information, see Create an IAM service role for a hybrid environment (p. 44).
- If you change the IAM instance profile, it might take some time for the instance credentials to refresh. SSM Agent will not process requests until this happens. To speed up the refresh process, you can restart SSM Agent or restart the instance.

About policies for a Systems Manager instance profile

This section describes the policies you can add to your EC2 instance profile for Systems Manager. To provide permissions for communication between instances and the Systems Manager API, we recommend creating custom policies that take into account your system needs and security requirements. However, as a starting point, you can use one or more of the following policies to grant permission for Systems Manager to interact with your instances. The first policy, AmazonSSMManagedInstanceCore, enables an instance to use AWS Systems Manager service core functionality. Depending on your operations plan, you might need permissions represented in one or more of the other three policies.

Policy: AmazonSSMManagedInstanceCore

Required permissions.
This AWS managed policy enables an instance to use Systems Manager service core functionality.

**Policy: A custom policy for S3 bucket access**

Required permissions in either of the following cases:

- **Case 1:** You are using a VPC endpoint to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink.

SSM Agent is Amazon software that is installed on your instances and performs Systems Manager tasks. This agent requires access to specific Amazon-owned S3 buckets. These buckets are publicly accessible.

In a private VPC endpoint environment, however, you must explicitly provide access to these buckets:

```
arn:aws:s3:::patch-baseline-snapshot-region/*
arwn:aws:s3:::aws-ssm-region/*
```

For more information, see Step 6: (Optional) Create a Virtual Private Cloud endpoint (p. 37), About minimum S3 Bucket permissions for SSM Agent (p. 95), and VPC Endpoints in the Amazon VPC User Guide.

- **Case 2:** You plan to use an S3 bucket that you create as part of your Systems Manager operations.

Your EC2 instance profile for Systems Manager must grant access to an S3 bucket that you own for tasks like the following:

- To access scripts you store in the S3 bucket to use in commands you run.
- To store the full output of Run Command commands or Session Manager sessions.
- To access custom patch lists for use when patching your instances.

**Note**

Saving output log data in an S3 bucket is optional, but we recommend setting it up at the beginning of your Systems Manager configuration process if you have decided to do so. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide.

**Policy: AmazonSSMDirectoryServiceAccess**

Required only if you plan to join EC2 instance for Windows Server to a Microsoft AD directory.

This AWS managed policy allows SSM Agent to access AWS Directory Service on your behalf for requests to join the domain by the managed instance. For more information, see Seamlessly Join a Windows EC2 Instance in the AWS Directory Service Administration Guide.

**Policy: CloudWatchAgentServerPolicy**

Required only if you plan to install and run the CloudWatch agent on your instances to read metric and log data on an instance and write it to Amazon CloudWatch. These help you monitor, analyze, and quickly respond to issues or changes to your AWS resources.

Your instance profile needs this policy only if you will use CloudWatch features, such as CloudWatch Events or CloudWatch Logs. (You can also create a more restrictive policy that, for example, limits writing access to a specific CloudWatch Logs log stream.)

**Note**

Using CloudWatch features is optional, but we recommend setting them up at the beginning of your Systems Manager configuration process if you have decided to use them. For more information, see the Amazon CloudWatch Events User Guide and the Amazon CloudWatch Logs User Guide.

To create an instance profile with permissions for additional Systems Manager services, see the following resources.
Task 1: (Optional) Create a custom policy for S3 bucket access

Creating a custom policy for Amazon S3 access is required only if you are using a VPC endpoint or using an S3 bucket of your own in your Systems Manager operations.

For information about the AWS managed S3 buckets you provide access to in the policy below, see About minimum S3 Bucket permissions for SSM Agent (p. 95).

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy.
3. Choose the JSON tab, and replace the default text with the following:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::aws-ssm-region/*",
                "arn:aws:s3:::aws-windows-downloads-region/*",
                "arn:aws:s3:::amazon-ssm-region/*",
                "arn:aws:s3:::amazon-ssm-packages-region/*",
                "arn:aws:s3:::region-birdwatcher-prod/*",
                "arn:aws:s3:::patch-baseline-snapshot-region/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetEncryptionConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-name/*",
                "arn:aws:s3:::my-bucket-name"
            ]
        }
    ]
}
```

1 The first Statement element is required only if you are using a VPC endpoint.

2 The second Statement element is required only if you are using an S3 bucket that you created to use in your Systems Manager operations.
The PutObjectAcl access control list permission is required only if you plan to support cross-account access to S3 buckets in other accounts.

The GetEncryptionConfiguration element is required if your S3 bucket is configured to use encryption.

If your S3 bucket is configured to use encryption, then the S3 bucket root (for example, arn:aws:s3:::my-bucket-name) must be listed in the Resource section. Your IAM user, group, or role must be configured with access to the root bucket.

If you are using a VPC endpoint in your operations, do the following:

In the first Statement element, replace each region placeholder with the identifier of the AWS Region this policy will be used in. For example, use us-east-2 for the US East (Ohio) Region. For a list of supported region values, see the Region column in Systems Manager service endpoints in the Amazon Web Services General Reference.

Important
We recommend that you avoid using wildcard characters (*) in place of specific Regions in this policy. For example, use arn:aws:s3:::aws-ssm-us-east-2/* and do not use arn:aws:s3:::aws-ssm-*/*. Using wildcards could provide access to S3 buckets that you don’t intend to grant access to. If you want to use the instance profile for more than one Region, we recommend repeating the first Statement element for each Region.

-or-

If you are not using a VPC endpoint in your operations, you can delete the first Statement element.

If you are using an S3 bucket of your own in your Systems Manager operations, do the following:

In the second Statement element, replace my-bucket-name with the name of an S3 bucket in your account. You will use this bucket for your Systems Manager operations. It provides permission for objects in the bucket, using *arn:aws:s3:::my-bucket-name/* as the resource. For more information about providing permissions for buckets or objects in buckets, see the topic Specifying Permissions in a Policy in the Amazon Simple Storage Service Developer Guide and the AWS blog post IAM Policies and Bucket Policies and ACLs! Oh, My! (Controlling Access to S3 Resources).

Note
If you use more than one bucket, provide the ARN for each one. For example, for permissions on buckets:

"Resource": [
  "arn:aws:s3:::my-first-bucket-name/*",
  "arn:aws:s3:::my-second-bucket-name/*"
]

-or-

If you are not using an S3 bucket of your own in your Systems Manager operations, you can delete the second Statement element.

Choose Review policy.

For Name, enter a name to identify this policy, such as SSMInstanceProfileS3Policy or another name that you prefer.

Choose Create policy.
Task 2: Add permissions to a Systems Manager instance profile (console)

Depending on whether you are creating a new role for your instance profile or adding the necessary permissions to an existing role, use one of the following procedures.

To create an instance profile for Systems Manager managed instances (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
3. Under Select type of trusted entity, choose AWS service.
4. Immediately under Choose the service that will use this role, choose EC2, and then choose Next: Permissions.

5. On the Attach permissions policies page, do the following:
   - Use the Search field to locate the AmazonSSMManagedInstanceCore. Select the box next to its name.
   - If you created a custom S3 bucket policy in the previous procedure, Task 1: (Optional) Create a custom policy for S3 bucket access (p. 32), search for it and select the box next to its name.
   - If you plan to join instances to an Active Directory managed by AWS Directory Service, search for AmazonSSMDirectoryServiceAccess and select the box next to its name.
   - If you plan to use CloudWatch Events or CloudWatch Logs to manage or monitor your instance, search for CloudWatchAgentServerPolicy and select the box next to its name.

6. Choose Next: Tags.
7. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose Next: Review.
8. For Role name, enter a name for your new instance profile, such as SSMInstanceProfile or another name that you prefer.
Step 5: Attach an IAM instance profile to an EC2 instance

SSM Agent requirements for instances

AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure these resources.

If the Amazon Machine Image (AMI) type you choose in the first procedure doesn't come with SSM Agent preinstalled, you must manually install the agent on the new instance before it can be used with Systems Manager. If SSM Agent isn't installed on the existing EC2 instance you choose in the second procedure, you must manually install the agent on the instance before it can be used with Systems Manager.

SSM Agent is installed by default on the following AMIs:

- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
Launch an instance that uses the Systems Manager instance profile (console)

To launch an instance that uses the Systems Manager instance profile (console)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation bar at the top of the screen, select the AWS Region for the instance.
3. Choose Launch Instance.
4. On the Choose an Amazon Machine Image (AMI) page, locate the AMI for the instance type you want to create, and then choose Select.
5. Choose the type of instance to launch, such as t2.micro, and then choose Next: Configure Instance Details.
6. On the Configure Instance Details page, in the IAM role drop-down list, select the instance profile you created using the procedure in Step 4: Create an IAM instance profile for Systems Manager (p. 30).
7. For other options on the page, make selections that meet your requirements for the instance. For more information, choose one of the following, depending on your selected operating system type:
   - Linux: Launching an Instance Using the Launch Instance Wizard in the Amazon EC2 User Guide for Linux Instances
   - Windows Server: Launching an Instance Using the Launch Instance Wizard in the Amazon EC2 User Guide for Windows Instances
8. Complete the wizard.

If you create other instances that you want to configure using Systems Manager, you must specify the instance profile for each instance.
Attach the Systems Manager instance profile to an existing instance (console)

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Instances**, choose **Instances**.
3. Browse to and choose your EC2 instance from the list.
4. In the **Actions** menu, choose **Instance Settings**, **Attach/Replace IAM Role**.
5. For **IAM role**, select the instance profile you created using the procedure in Step 4: Create an IAM instance profile for Systems Manager (p. 30).
6. Choose **Apply**.

For more information about attaching IAM roles to instances, see Attaching an IAM Role to an Instance in the Amazon EC2 User Guide for Linux Instances.

Continue to Step 6: (Optional) Create a Virtual Private Cloud endpoint (p. 37).

Step 6: (Optional) Create a Virtual Private Cloud endpoint

You can improve the security posture of your managed instances (including managed instances in your hybrid environment) by configuring AWS Systems Manager to use an interface VPC endpoint in Amazon Virtual Private Cloud (Amazon VPC). An interface VPC endpoint (interface endpoint) enables you to connect to services powered by AWS PrivateLink, a technology that enables you to privately access Amazon EC2 and Systems Manager APIs by using private IP addresses. PrivateLink restricts all network traffic between your managed instances, Systems Manager, and Amazon EC2 to the Amazon network. (Managed instances don't have access to the Internet.) Also, you don’t need an Internet gateway, a NAT device, or a virtual private gateway.

You are not required to configure PrivateLink, but it's recommended. For more information about PrivateLink and VPC endpoints, see Accessing AWS Services Through PrivateLink.

**Note**

The alternative to using a VPC endpoint is to enable outbound internet access on your managed instances. In this case, the managed instances must also allow HTTPS (port 443) outbound traffic to the following endpoints:

- `ssm.region.amazonaws.com`
- `ssmmessages.region.amazonaws.com`
- `ec2messages.region.amazonaws.com`

For more information about calls to these endpoints, see Reference: ec2messages, ssmmessages, and other API calls (p. 1195).

About Amazon VPC

Amazon Virtual Private Cloud (Amazon VPC) enables you to define a virtual network in your own logically isolated area within the AWS cloud, known as a virtual private cloud (VPC). You can launch your AWS resources, such as instances, into your VPC. Your VPC closely resembles a traditional network that you might operate in your own data center, with the benefits of using AWS's scalable infrastructure. You can configure your VPC; you can select its IP address range, create subnets, and configure route tables,
network gateways, and security settings. You can connect instances in your VPC to the internet. You can connect your VPC to your own corporate data center, making the AWS cloud an extension of your data center. To protect the resources in each subnet, you can use multiple layers of security, including security groups and network access control lists. For more information, see the Amazon VPC User Guide.

Topics
- VPC endpoint restrictions and limitations (p. 38)
- Creating VPC endpoints for Systems Manager (p. 39)

VPC endpoint restrictions and limitations

Before you configure VPC endpoints for Systems Manager, be aware of the following restrictions and limitations.

aws:domainJoin plugin

If you choose to create VPC endpoints, then be aware that requests to join a Windows Server instance to a domain from SSM documents that use the `aws:domainJoin` plugin will fail. This plugin requires the AWS Directory Service, and AWS Directory Service does not have PrivateLink endpoint support. Support for joining a Windows Server instance to a domain from other domain join methods depend only on Active Directory requirements (for example, ensuring that domain controllers are reachable and discoverable by using DNS and other related requirements). You can use Amazon EC2 User Data scripts to join an instance to a domain.

Cross-region requests

VPC endpoints currently do not support cross-region requests—ensure that you create your endpoint in the same region as your bucket. You can find the location of your bucket by using the Amazon S3 console, or by using the `get-bucket-location` command. Use a Region-specific Amazon S3 endpoint to access your bucket; for example, `mybucket.s3-us-west-2.amazonaws.com`. For more information about Region-specific endpoints for Amazon S3, see Amazon S3 Service Endpoints in the Amazon Web Services General Reference. If you use the AWS CLI to make requests to Amazon S3, set your default region to the same region as your bucket, or use the `--region` parameter in your requests.

VPC Peering connections

VPC interface endpoints can be accessed through both `intra-Region` and `inter-Region` VPC peering connections. For more information about VPC peering connection requests for VPC interface endpoints, see Interface Endpoints Properties and Limitations in the Amazon Virtual Private Cloud User Guide.

VPC Gateway Endpoint connections cannot be extended out of a VPC. Resources on the other side of a VPC peering connection in your VPC cannot use the gateway endpoint to communicate with resources in the gateway endpoint service. For more information about VPC peering connection requests for VPC gateway endpoints, see Gateway Endpoint Limitations in the Amazon Virtual Private Cloud User Guide.

Incoming connections

The security group attached to the VPC endpoint must allow incoming connections on port 443 from the private subnet of the managed instance. If incoming connections are not allowed, then the managed instance cannot connect to the SSM and EC2 endpoints.

Amazon S3 buckets

Your VPC endpoint policy must allow at least access to the following Amazon S3 buckets:

- The S3 buckets used by Patch Manager for patch baseline operations in your AWS Region. These buckets contain the code that is retrieved and run on instances by the patch baseline service. Each AWS Region has its own patch baseline operations buckets for the code to be retrieved when a patch baseline document is run. If the code can't be downloaded, the patch baseline command will fail.
To provide access to the buckets in your AWS Region, include the following permission in your endpoint policy:

**Note**
In the Middle East (Bahrain) Region (me-south-1) only, these buckets use different naming conventions. For this AWS Region only, use the following two buckets instead.

- `patch-baseline.snapshot-me-south-1-uduvl7q8`
- `aws-patch-manager-me-south-1-a53fc9dce`

```
arn:aws:s3:::patch-baseline-snapshot-region/*
ar:aws:s3:::aws-ssm-region/*
```

*region* represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported *region* values, see the **Region** column in **Systems Manager service endpoints** in the *Amazon Web Services General Reference*.

For example:

```
arn:aws:s3:::patch-baseline-snapshot-us-east-2/*
ar:aws:s3:::aws-ssm-us-east-2/*
```

- The S3 buckets listed in *About minimum S3 Bucket permissions for SSM Agent* (p. 95).

**DNS in hybrid environment**

For information about configuring DNS to work with PrivateLink endpoints in hybrid environments, see *Private DNS*. If you want to use your own DNS, you can use Route 53 Resolver. For more information, see *Resolving DNS Queries Between VPCs and Your Network* in the *Amazon Route 53 Developer Guide*.

**Creating VPC endpoints for Systems Manager**

Use the following information to create VPC interface and gateway endpoints for Systems Manager. This topic links to procedures in the *Amazon VPC User Guide*.

**To create VPC endpoints for Systems Manager**

In the first step below, you create three required and one optional *interface* endpoints for Systems Manager. The first three endpoints are required for Systems Manager to work in a VPC. The fourth, `com.amazonaws.region.ssmmessages`, is required only if you are using Session Manager capabilities.

In the second step, you create the required *gateway* endpoint for Systems Manager to access Amazon S3.

**Note**

*region* represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported *region* values, see the **Region** column in **Systems Manager service endpoints** in the *Amazon Web Services General Reference*.

1. Follow the steps in *Creating an Interface Endpoint* to create the following interface endpoints:

   - `com.amazonaws.region.ssm`: The endpoint for the Systems Manager service.
   - `com.amazonaws.region.ec2messages`: Systems Manager uses this endpoint to make calls from SSM Agent to the Systems Manager service.
   - `com.amazonaws.region.ec2`: If you're using Systems Manager to create VSS-enabled snapshots, you need to ensure that you have an endpoint to the EC2 service. Without the EC2 endpoint
defined, a call to enumerate attached EBS volumes fails, which causes the Systems Manager command to fail.

- **com.amazonaws.region.ssmmessages**: This endpoint is required only if you are connecting to your instances through a secure data channel using Session Manager. For more information, see AWS Systems Manager Session Manager (p. 770) and Reference: ec2messages, ssmmessages, and other API calls (p. 1195).

2. Follow the steps in Creating a Gateway Endpoint to create the following gateway endpoint for Amazon S3.

- **com.amazonaws.region.s3**: Systems Manager uses this endpoint to update SSM Agent and for tasks like uploading output logs you choose to store in S3 buckets, retrieving scripts or other files you store in buckets, and so on.

Continue to Step 7: (Optional) Create Systems Manager service roles (p. 40).

### Step 7: (Optional) Create Systems Manager service roles

This topic explains the difference between a *service role* and a *service-linked role* for Systems Manager. It also explains when you need to create or use either type of role.

**Service role**: A service role is an AWS Identity and Access Management (IAM) that grants permissions to an AWS service so that the service can access AWS resources. Only a few Systems Manager scenarios require a service role. When you create a service role for Systems Manager, you choose the permissions to grant in order for it to access or interact with other AWS resources.

**Service-linked role**: A service-linked role is predefined by Systems Manager and includes all the permissions that the service requires to call other AWS services on your behalf.

Currently, the Systems Manager service-linked role can be used for the following:

- The Systems Manager Inventory capability uses the service-linked role to collect inventory metadata from tags and resource groups.
- The Maintenance Windows capability can use the service-linked role in some situations. Other situations require a custom service role that you create, as described below.

For more information about the service-linked role, see Using service-linked roles for Systems Manager (p. 1122).

#### Create a service role

You can create the following service roles as part of Systems Manager setup, or you can create them later.

**Service role for Automation**

Automation previously required that you specify a service role so that the service had permission to perform actions on your behalf. Automation no longer requires this role because the service now operates by using the context of the user who invoked the execution.

However, the following situations still require that you specify a service role for Automation:
When you want to restrict a user’s privileges on a resource, but you want the user to run an Automation workflow that requires elevated privileges. In this scenario, you can create a service role with elevated privileges and allow the user to run the workflow.

Operations that you expect to run longer than 12 hours require a service role.

If you need to create a service role and an instance profile role for Automation, you can use one of the following methods.

- Method 1: Use AWS CloudFormation to configure a service role for Automation (p. 281)
- Method 2: Use IAM to configure roles for Automation (p. 283)

Service role for maintenance window tasks

To run tasks on your managed instances, the Maintenance Windows service must have permission to access those resources. This permission can be granted using either a service-linked role for Systems Manager or a custom service role that you create.

You create a custom service role in the following cases:

- If you want to use a more restrictive set of permissions than those provided by the service-linked role.
- If you need a more permissive or expanded set of permissions than those provided by the service-linked role. For example, some actions in Automation documents require permissions for actions in other AWS services.

For more information, see the following topics in the Maintenance Windows section of this user guide:

- Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622)
- (Optional) Create a custom service role for Maintenance Windows (Console) (p. 623).

Service role for Amazon Simple Notification Service notifications

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. In Systems Manager, you can configure Amazon SNS to send notifications about the status of commands that you send using the Run Command capability, or the status of tasks run in maintenance windows.

You create a service role for Amazon SNS as part of the process of configuring the service for use with Systems Manager. After you complete this configuration, you choose whether to receive notifications for particular Run Command commands or maintenance windows tasks at the time you create each one.

For more information, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

Service role for a Systems Manager hybrid environment

If you plan to use Systems Manager to manage on-premises servers and virtual machines (VMs) in what is called a hybrid environment, you must create an IAM role for those resources to communicate with the Systems Manager service.

For more information, see Create an IAM service role for a hybrid environment (p. 44).

Continue to Step 8: (Optional) Set up integrations with other AWS services (p. 42).
Step 8: (Optional) Set up integrations with other AWS services

AWS Systems Manager integrates with a number of other AWS services. In most cases, you set up an integration after you decide to incorporate the service into your Systems Manager operations. For example:

- Referencing AWS Secrets Manager secrets from Parameter Store parameters (p. 98)
- Using Chef InSpec profiles with Systems Manager Compliance (p. 113)

You can use some AWS services immediately to compile log data for later troubleshooting and analysis. You can also use AWS services to monitor and quickly respond to changes in your Systems Manager environment. Therefore, we recommend that you set up the following resources as part of your initial Systems Manager setup process.

**Amazon CloudWatch Events** and **Amazon Simple Notification Service** – CloudWatch Events lets you set up rules to detect when changes happen to AWS resources that you specify. You can configure CloudWatch Events to log status execution changes of the commands users in your account send using Systems Manager. You can create a rule to detect when a user in your organization starts or stops a session in Session Manager. You can also configure a CloudWatch event to trigger other actions in your AWS environment. For more information, see the following topics:

- Understanding command statuses (p. 845)
- Configuring CloudWatch Events for Run Command (p. 1149)
- Monitoring session activity using Amazon CloudWatch Events (console) (p. 826)

**Amazon Simple Storage Service (Amazon S3)**

Run Command command output in the Systems Manager console is truncated after 2,500 characters. In order to access complete command output logs, you can store Systems Manager output in an Amazon Simple Storage Service (Amazon S3) bucket, and then use this output later for auditing or troubleshooting. You specify whether to save command output to an S3 bucket each time you run a command. You can also create an Amazon S3 key prefix (a subfolder) to help you organize the log output. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide.

**Amazon CloudWatch Logs (CloudWatch Logs)**

As an alternative to storing command output in an S3 bucket, you can send output to an Amazon CloudWatch Logs log group. If you specify CloudWatch Logs as the output target, Run Command periodically sends all command output and error logs to CloudWatch Logs. You can monitor output logs in near real-time, search for specific phrases, values, or patterns, and create alarms based on the search. For more information, see Configuring Amazon CloudWatch Logs for Run Command (p. 1146).
Setting up AWS Systems Manager for hybrid environments

This section describes the setup tasks that account and system administrators perform for a hybrid environment. A hybrid environment includes on-premises servers and virtual machines (VMs) that have been configured for use with Systems Manager, including VMs in other cloud environments. After these steps are complete, users who have been granted permissions by the AWS account administrator can use AWS Systems Manager to configure and manage their organization’s on-premises servers and virtual machines (VMs).

If you plan to use Systems Manager to manage Amazon Elastic Compute Cloud (EC2) instances, or to use both EC2 instances and your own resources in a hybrid environment, follow the steps in Setting up AWS Systems Manager (p. 25) first.

Configuring your hybrid environment for Systems Manager enables you to do the following:

• Create a consistent and secure way to remotely manage your hybrid workloads from one location using the same tools or scripts.

• Centralize access control for actions that can be performed on your servers and VMs by using AWS Identity and Access Management (IAM).

• Centralize auditing and your view into the actions performed on your servers and VMs by recording all actions in AWS CloudTrail.

For information about using CloudTrail to monitor Systems Manager actions, see Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144).

• Centralize monitoring by configuring CloudWatch Events and Amazon SNS to send notifications about service execution success.

For information about using CloudWatch Events to monitor Systems Manager events, see Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147).

About managed instances

After you finish configuring your servers and VMs for Systems Manager as described in this section, your hybrid machines are listed in the AWS Management Console and described as managed instances. EC2 instances configured for Systems Manager are also described as managed instances. In the console, however, the IDs of your hybrid instances are distinguished from EC2 instances with the prefix “mi-”. EC2 instance IDs use the prefix “i-”.

For more information, see AWS Systems Manager Managed Instances (p. 758).

About instance tiers

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers and VMs in your hybrid environment. The standard-instances tier enables you to register a maximum of 1,000 on-premises servers or VMs per AWS account per AWS Region. If you need to register more than 1,000 on-premises servers or VMs in a single account and Region, then use the advanced-instances tier. Advanced instances also enable you to connect to your hybrid machines by using AWS Systems Manager Session Manager. Session Manager provides interactive shell access to your instances.
Step 1: Complete general Systems Manager setup steps

If you haven’t already done so, complete the following general setup steps for Systems Manager in the Setting up AWS Systems Manager (p. 25) section of this user guide. The other steps in that section are required only if you plan to manage EC2 instances.

- Sign up for AWS (p. 25)
- Create an Admin IAM user for AWS (p. 26)
- Create non-Admin IAM users and groups for Systems Manager (p. 27)
- (Optional) Create a Virtual Private Cloud endpoint (p. 37)
- (Optional) Create Systems Manager service roles (p. 40)
- (Optional) Set Up integrations with other AWS services (p. 42)

After ensuring that you have completed those steps, continue to Step 2: Create an IAM service role for a hybrid environment (p. 44).

Step 2: Create an IAM service role for a hybrid environment

Servers and virtual machines (VMs) in a hybrid environment require an IAM role to communicate with the Systems Manager service. The role grants AssumeRole trust to the Systems Manager service. You only need to create the service role for a hybrid environment once for each AWS account.

**Note**

Users in your company or organization who will use Systems Manager on your hybrid machines must be granted permission in IAM to call the SSM API. For more information, see Create non-Admin IAM users and groups for Systems Manager (p. 27).

**S3 bucket policy requirement**

If either of the following cases are true, you must create a custom IAM permission policy for S3 buckets before completing this procedure:

- **Case 1**: You are using a VPC endpoint to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink.
- **Case 2**: You plan to use an S3 bucket that you create as part of your Systems Manager operations, such as for storing output for Run Command commands or Session Manager sessions to an S3 bucket.
Before proceeding, follow the steps in Create a custom S3 bucket policy for an instance profile (p. 32). The information about S3 bucket policies in that topic also applies to your service role.

**AWS CLI**

**To create an IAM service role for a hybrid environment (AWS CLI)**

1. Create a text file with a name such as `SSMService-Trust.json` with the following trust policy. Make sure to save the file with the `.json` file extension.

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

2. Use the `create-role` command to create the service role. This example creates a role named `SSMServiceRole`. You can choose another name if you prefer.

   **Linux**
   ```bash
   aws iam create-role
   --role-name SSMServiceRole
   --assume-role-policy-document file://SSMService-Trust.json
   ```

   **Windows**
   ```bash
   aws iam create-role ^
   --role-name SSMServiceRole ^
   --assume-role-policy-document file://SSMService-Trust.json
   ```

3. Use `attach-role-policy` as follows to enable the service role you just created to create a session token. The session token gives your managed instance permission to run commands using Systems Manager.

   **Note**
   The policies you add for a service profile for managed instances in a hybrid environment are the same policies used to create an instance profile for EC2 instances. For more information about the AWS policies used in the following commands, see Create an IAM instance profile for Systems Manager (p. 30).

   (Required) Run the following command to enable a managed instance to use AWS Systems Manager service core functionality.

   **Linux**
   ```bash
   aws iam attach-role-policy
   --role-name SSMServiceRole
   --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
   ```

   **Windows**
   ```bash
   aws iam attach-role-policy ^
   --role-name SSMServiceRole ^
   --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
   ```
If you created a custom S3 bucket policy for your service role, run the following command to enable SSM Agent to access the buckets you specified in the policy. Replace `account-id` and `my-bucket-policy-name` with your AWS account ID and your bucket name.

**Linux**

```bash
aws iam attach-role-policy \
    --role-name SSMServiceRole \
    --policy-arn arn:aws:iam::account-id:policy/my-bucket-policy-name
```

**Windows**

```bash
aws iam attach-role-policy ^
    --role-name SSMServiceRole ^
    --policy-arn arn:aws:iam::account-id:policy/my-bucket-policy-name
```

(Optional) Run the following command to allow SSM Agent to access AWS Directory Service on your behalf for requests to join the domain by the managed instance. Your instance profile needs this policy only if you join your instances to a Microsoft AD directory.

**Linux**

```bash
aws iam attach-role-policy \
    --role-name SSMServiceRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonSSMDirectoryServiceAccess
```

**Windows**

```bash
aws iam attach-role-policy ^
    --role-name SSMServiceRole ^
    --policy-arn arn:aws:iam::aws:policy/AmazonSSMDirectoryServiceAccess
```

(Optional) Run the following command to allow the CloudWatch agent to run on your managed instances. This command makes it possible to read information on an instance and write it to CloudWatch. Your service profile needs this policy only if you will use CloudWatch features, such as Amazon CloudWatch Events or Amazon CloudWatch Logs.

```bash
aws iam attach-role-policy \
    --role-name SSMServiceRole \
    --policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```

**Tools for PowerShell**

**To create an IAM service role for a hybrid environment (AWS Tools for Windows PowerShell)**

1. Create a text file with a name such as `SSMService-Trust.json` with the following trust policy. Make sure to save the file with the `.json` file extension.

```json
{
    "Version": "2012-10-17",
    "Statement": {
```
2. Use **New-IAMRole** as follows to create a service role. This example creates a role named **SSMServiceRole**. You can choose another name if you prefer.

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

3. Use **Register-IAMRolePolicy** as follows to enable the service role you created to create a session token. The session token gives your managed instance permission to run commands using Systems Manager.

   **Note**
   The policies you add for a service profile for managed instances in a hybrid environment are the same policies used to create an instance profile for EC2 instances. For more information about the AWS policies used in the following commands, see Create an IAM instance profile for Systems Manager (p. 30).

   (Required) Run the following command to enable a managed instance to use AWS Systems Manager service core functionality.

```bash
Register-IAMRolePolicy -RoleName SSMServiceRole -PolicyArn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

   If you created a custom S3 bucket policy for your service role, run the following command to enable SSM Agent to access the buckets you specified in the policy. Replace `account-id` and `my-bucket-policy-name` with your AWS account ID and your bucket name.

```bash
Register-IAMRolePolicy -RoleName SSMServiceRole -PolicyArn arn:aws:iam::account-id:policy/my-bucket-policy-name
```

   (Optional) Run the following command to allow SSM Agent to access AWS Directory Service on your behalf for requests to join the domain by the managed instance. Your instance profile needs this policy only if you join your instances to a Microsoft AD directory.

```bash
Register-IAMRolePolicy -RoleName SSMServiceRole -PolicyArn arn:aws:iam::aws:policy/AmazonSSMDirectoryServiceAccess
```

   (Optional) Run the following command to allow the CloudWatch agent to run on your managed instances. This command makes it possible to read information on an instance and write it to CloudWatch. Your service profile needs this policy only if you will use CloudWatch features, such as Amazon CloudWatch Events or Amazon CloudWatch Logs.

```bash
Register-IAMRolePolicy -RoleName SSMServiceRole -PolicyArn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
```
Step 3: Install a TLS certificate on on-premises servers and VMs

A Transport Layer Security (TLS) certificate must be installed on each managed instance you use with Systems Manager. AWS services use these certificates to encrypt calls to other AWS services.

A TLS certificate is already installed by default on each EC2 instance created from any Amazon Machine Image (AMI). Most modern operating systems include the required TLS certificate from Amazon Trust Services CAs in their trust store.

If you find the required Amazon Trust Services CA certificates are not installed on your base operating systems, on instances created from AMIs that are not supplied by Amazon, or on your own on-premises servers and VMs, you must install and enable a certificate from Amazon Trust Services, or use AWS Certificate Manager (ACM) to create and manage certificates for a supported integrated service.

Each of your managed instances must have one of the following Transport Layer Security (TLS) certificates installed.

- Amazon Root CA 1
- Starfield Services Root Certificate Authority - G2
- Starfield Class 2 Certificate Authority

For information about using ACM, see the AWS Certificate Manager User Guide. If certificates in your computing environment are managed by a Group Policy Object (GPO), then you might need to configure Group Policy to include one of these certificates.

For more information about the Amazon Root and Starfield certificates, see the blog post How to Prepare for AWS’s Move to Its Own Certificate Authority.

Continue to Step 4: Create a managed-instance activation for a hybrid environment (p. 48).

Step 4: Create a managed-instance activation for a hybrid environment

To set up servers and virtual machines (VMs) in your hybrid environment as managed instances, you need to create a managed-instance activation. After you successfully complete the activation, you immediately receive an Activation Code and Activation ID. You specify this Code/ID combination when you install SSM Agent on servers and VMs in your hybrid environment. The Code/ID provides secure access to the Systems Manager service from your managed instances.

Important

Systems Manager immediately returns the Activation Code and ID to the console or the command window, depending on how you created the activation. Copy this information and store it in a safe place. If you navigate away from the console or close the command window, you might lose this information. If you lose it, you must create a new activation.
About activation expirations

An activation expiration is a window of time when you can register on-premises machines with Systems Manager. An expired activation has no impact on your servers or virtual machines (VMs) that you previously registered with Systems Manager. If an activation expires then you can’t register more servers or VMs with Systems Manager by using that specific activation. You simply need to create a new one.

Every on-premises server and VM you previously registered remains registered as a Systems Manager managed instance until you explicitly deregister it. You can deregister a managed instance on the Managed Instances page of the Systems Manager console, by using the AWS CLI command deregister-managed-instance, or by using the API action DeregisterManagedInstance.

About activation tags

If you create an activation by using either the AWS CLI or AWS Tools for Windows PowerShell, you can specify tags. Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. Here is an AWS CLI sample command to run on a local Linux machine that includes tags.

```bash
aws ssm create-activation
  --default-instance-name MyWebServers 
  --description "Activation for Finance department webservers"
  --iam-role service-role/AmazonEC2RunCommandRoleForManagedInstances 
  --registration-limit 10 
  --region us-east-2 
  --tags "Key=Department,Value=Finance"
```

If you specify tags when you create an activation, then those tags are automatically assigned to your on-premises servers and VMs when you activate them.

You can't add tags to or delete tags from an existing activation. If you don't want to automatically assign tags to your on-premises servers and VMs using an activation, then you can add tags to them later. More specifically, you can tag your on-premises servers and VMs after they connect to Systems Manager for the first time. After they connect, they are assigned a managed instance ID and listed in the Systems Manager console with an ID that is prefixed with "mi-". For information about how to add tags to your managed instances without using the activation process, see Tagging managed instances (p. 1172).

Note

You can't assign tags to an activation if you create it by using the Systems Manager console. You must create it by using either the AWS CLI or Tools for Windows PowerShell.

If you no longer want to manage an on-premises server or virtual machine (VM) by using Systems Manager, you can deregister it. For information, see Deregistering managed instances in a hybrid environment (p. 769).

Topics

- Create an activation (console) (p. 49)
- Create a managed instance activation (command line) (p. 50)

Create an activation (console)

To create a managed-instance activation

2. In the navigation pane, choose Hybrid Activations.

- or -
Create a managed instance activation (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to create a managed instance activation.

To create an activation

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create an activation.
   
   **Note**
   
   region represents the identifier for an AWS Region supported by AWS Systems Manager, such as us-east-2 for the US East (Ohio) Region. For a list of supported region values, see the Region column in Systems Manager service endpoints in the Amazon Web Services General Reference.

   **Linux**

   ```bash
   aws ssm create-activation \ 
   --default-instance-name name \ 
   --iam-role iam-service-role-name \ 
   --registration-limit number-of-managed-instances \ 
   --region region \ 
   --tags "Key=key-name-1,Value=key-value-1" "Key=key-name-2,Value=key-value-2"
   ```

   **Windows**

   ```bash
   aws ssm create-activation ^
   ```
Create a managed instance activation (command line)

```
--default-instance-name name ^
--iam-role iam-service-role-name ^
--registration-limit number-of-managed-instances ^
--region region ^
--tags "Key=key-name-1,Value=key-value-1" "Key=key-name-2,Value=key-value-2"
```

PowerShell

```
New-SSMActivation -DefaultInstanceName name `  
-IamRole iam-service-role-name `  
-RegistrationLimit number-of-managed-instances `  
-Region region `  
-Tag @{"Key"="key-name-1";"Value"="key-value-1"},@{"Key"="key- 
-name-2";"Value"="key-value-2"}
```

Here is an example.

Linux

```
aws ssm create-activation \
   --default-instance-name MyWebServers \
   --iam-role service-role/AmazonEC2RunCommandRoleForManagedInstances \
   --registration-limit 10 \
   --region us-east-2 \
   --tags "Key=Environment,Value=Production" "Key=Department,Value=Finance"
```

Windows

```
aws ssm create-activation ^
   --default-instance-name MyWebServers ^
   --iam-role service-role/AmazonEC2RunCommandRoleForManagedInstances ^
   --registration-limit 10 ^
   --region us-east-2 ^
   --tags "Key=Environment,Value=Production" "Key=Department,Value=Finance"
```

PowerShell

```
New-SSMActivation -DefaultInstanceName MyWebServers `  
-IamRole service-role/AmazonEC2RunCommandRoleForManagedInstances `  
-RegistrationLimit 10 `  
-Region us-east-2 `  
-Tag @{"Key"="Environment";"Value"="Production"},@{"Key"="Department";"Value"="Finance"}
```

If the activation is created successfully, the system immediately returns an Activation Code and ID.

Continue to Step 6: Install SSM Agent for a hybrid environment (Windows) (p. 55).
Step 5: Install SSM Agent for a hybrid environment (Linux)

This topic describes how to install SSM Agent on Linux machines in a hybrid environment. If you plan to use Windows Server machines in a hybrid environment, see the previous step, Step 6: Install SSM Agent for a hybrid environment (Windows) (p. 55).

Important
This procedure is for servers and virtual machines (VMs) in an on-premises or hybrid environment. To download and install SSM Agent on an EC2 instance for Linux, see Installing and configuring SSM Agent on EC2 instances for Linux (p. 69).

Before you begin, locate the Activation Code and Activation ID that were sent to you after you completed the managed-instance activation earlier in Step 4: Create a managed-instance activation for a hybrid environment (p. 48). You specify the Code and ID in the following procedure.

The URLs in the following scripts let you download SSM Agent from any AWS region. If you want to download the agent from a specific region, copy the URL for your operating system, and then replace \texttt{region} with an appropriate value.

\texttt{region} represents the identifier for an AWS Region supported by AWS Systems Manager, such as \texttt{us-east-2} for the US East (Ohio) Region. For a list of supported \texttt{region} values, see the \texttt{Region} column in Systems Manager service endpoints in the Amazon Web Services General Reference.

For example, to download SSM Agent for Amazon Linux, RHEL, CentOS, and SLES 64-bit from the US West (N. California) Region (us-west-1), use the following URL:

\begin{verbatim}
\end{verbatim}

Amazon Linux 2, Amazon Linux, RHEL, Oracle Linux, CentOS, and SLES

- **64-bit**
  
  \begin{verbatim}
  \end{verbatim}

- **32-bit**
  
  \begin{verbatim}
  \end{verbatim}

Ubuntu Server

- **64-bit**
  
  \begin{verbatim}
  \end{verbatim}

- **32-bit**
  
  \begin{verbatim}
  \end{verbatim}

Debian Server

- **64-bit**
To install SSM Agent on servers and VMs in your hybrid environment

1. Log on to a server or VM in your hybrid environment.
2. Copy and paste one of the following command blocks into SSH. Replace the placeholder values with the Activation Code and Activation ID generated when you create a managed-instance activation, and with the identifier of the AWS Region you want to download SSM Agent from.

Note that `sudo` is not necessary if you are a root user.

Raspbian


**Amazon Linux, RHEL 6.x, and CentOS 6.x**

```bash
mkdir /tmp/ssm
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
```

**Amazon Linux 2, RHEL 7.x, Oracle Linux, and CentOS 7.x**

```bash
mkdir /tmp/ssm
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
```

**RHEL 8.x**

```bash
mkdir /tmp/ssm
sudo dnf 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
```
## Step 5: Install SSM Agent for a hybrid environment (Linux)

### Debian

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

### Raspbian

```
mkdir /tmp/ssm
sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
sudo amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
sudo service amazon-ssm-agent start
```

### SLES

```
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
```

### Ubuntu

- **Using .deb packages**

  ```
  mkdir /tmp/ssm
  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
  ```

- **Using Snap packages**

  You don't need to specify a URL for the download, because the `snap` command automatically downloads the agent from the Snap app store at [https://snapcraft.io](https://snapcraft.io).

  ```
  sudo snap install amazon-ssm-agent --classic
  sudo systemctl stop snap.amazon-ssm-agent.amazon-ssm-agent.service
  sudo /snap/amazon-ssm-agent/current/amazon-ssm-agent -register -code "activation-code" -id "activation-id" -region "region"
sudo systemctl start snap.amazon-ssm-agent.amazon-ssm-agent.service
  ```
Step 6: Install SSM Agent for a hybrid environment (Windows)

This topic describes how to install SSM Agent on Windows Server machines in a hybrid environment. If you plan to use Linux machines in a hybrid environment, see the next step, Step 5: Install SSM Agent for a hybrid environment (Linux) (p. 52).

Important
This procedure is for servers and virtual machines (VMs) in an on-premises or hybrid environment. To download and install SSM Agent on an EC2 instance for Windows Server, see Installing and configuring SSM Agent on Windows Server instances (p. 65).

Before you begin, locate the Activation Code and Activation ID that were sent to you after you completed the managed-instance activation earlier in Step 4: Create a managed-instance activation for a hybrid environment (p. 48). You specify the Code and ID in the following procedure.

To install SSM Agent on servers and VMs in your hybrid environment

1. Log on to a server or VM in your hybrid environment.
2. Open Windows PowerShell in elevated (administrative) mode.
3. Copy and paste the following command block into AWS Tools for Windows PowerShell. Replace the placeholder values with the Activation Code and Activation ID generated when you create a managed-instance activation, and with the identifier of the AWS Region you want to download SSM Agent from.

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

Press Enter.

The command downloads and installs SSM Agent onto the server or VM in your hybrid environment. The command stops SSM Agent, and then registers the server or VM with the SSM service. The server or VM is now a managed instance. EC2 instances configured for Systems Manager are also managed instances. In the Systems Manager console, however, your on-premises instances are distinguished from EC2 instances with the prefix "mi-".

Note
You can deregister a managed instance by calling the DeregisterManagedInstance API action from either the AWS CLI or Tools for Windows PowerShell. Here’s an example CLI command:

```bash
aws ssm deregister-managed-instance --instance-id "mi-1234567890"
```

Continue to Step 6: Install SSM Agent for a hybrid environment (Windows) (p. 55).
region represents the identifier for an AWS Region supported by AWS Systems Manager, such as us-east-2 for the US East (Ohio) Region. For a list of supported region values, see the Region column in Systems Manager service endpoints in the Amazon Web Services General Reference.

```bash
#code = "activation-code"
$id = "activation-id"
$region = "region"
$dir = $env:TEMP + "\ssm"
New-Item -ItemType directory -Path $dir -Force
cd $dir
Start-Process .\AmazonSSMAgentSetup.exe -ArgumentList @("/q", "/log", "install.log", "CODE=$code", "ID=$id", "REGION=$region") -Wait
Get-Content ($env:ProgramData + "\Amazon\SSM\InstanceData\registration")
Get-Service -Name "AmazonSSMAgent"
```

4. Press Enter.

The command does the following:

- Downloads and installs SSM Agent onto the server or VM.
- Registers the server or VM with the SSM service.
- Returns a response to the request similar to the following:

```
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
```

The server or VM is now a managed instance. These instances are now identified with the prefix "mi-". You can view managed instances on the Managed Instances page in the Systems Manager console, by using the AWS CLI command describe-instance-information, or by using the API command DescribeInstancePatches.

**Note**
You can deregister a managed instance by calling the DeregisterManagedInstance API action from either the AWS CLI or Tools for Windows PowerShell. Here’s an example CLI command:

```
aws ssm deregister-managed-instance --instance-id "mi-1234567890"
```
Getting started with AWS Systems Manager

This section helps you learn about and use AWS Systems Manager after your organization completes the setup steps in Setting up AWS Systems Manager (p. 25) or Setting up AWS Systems Manager for hybrid environments (p. 43).

Before You Begin

The following is useful background information to help you get started:

- The topic What is AWS Systems Manager? (p. 1)
- The Amazon EC2 Getting Started Guide (if you are managing Amazon Elastic Compute Cloud (EC2) instances in your account).
- Understanding the Systems Manager setup requirements helps you troubleshoot problems you encounter while you use Systems Manager, such as with permissions or resource availability:
  - Setting up AWS Systems Manager (p. 25)
  - Setting up AWS Systems Manager for hybrid environments (p. 43)

When you are ready, continue with the following steps.

Topics

- Step 1: Install or upgrade AWS command line tools (p. 57)
- Step 2: Practice installing or updating SSM Agent on an instance (p. 59)
- Step 3: Try Systems Manager tutorials and walkthroughs (p. 60)

Step 1: Install or upgrade AWS command line tools

This topic is for users who have programmatic access to use Systems Manager (or any other AWS service), and who want to run AWS CLI or AWS Tools for Windows PowerShell commands from their local machines.

Note

Programmatic access and console access are different permissions that can be granted to a user account by an AWS account administrator. A user can be granted one or both access types. For information, see Create non-Admin IAM users and groups for Systems Manager (p. 27).

For information about the AWS CLI, see the AWS Command Line Interface User Guide. For information about the AWS Tools for Windows PowerShell, see the AWS Tools for Windows PowerShell User Guide.

For information about all Systems Manager commands you can run using the AWS CLI, see the Systems Manager section of the AWS CLI Command Reference. For information about all Systems Manager commands you can run using the AWS Tools for PowerShell, see the Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference.

Important

Beginning January 10th, 2020, AWS CLI version 1.17 and later will no longer support Python 2.6 or Python 3.3. After this date, the installer for the AWS CLI will require Python 2.7, Python 3.4,
or a later version to successfully install the AWS CLI. For more information, see Using the AWS CLI version 1 with Python 2.6 or Python 3.3 in the IAM User Guide.

AWS CLI

**To install or upgrade and then configure the AWS CLI**

1. Follow the instructions in Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide to install or upgrade the AWS CLI on your local machine.

   **Tip**
   The AWS CLI is frequently updated with new functionality. Upgrade (reinstall) the CLI periodically to ensure that you have access to all the latest functionality.

2. To configure the AWS CLI, see Configuring the AWS Command Line Interface in the AWS Command Line Interface User Guide.

In this step, you specify credentials that an AWS administrator in your organization has given you, in the following format:

AWS Access Key ID: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

**Important**
When you configure the AWS CLI, you are prompted to specify an AWS Region. Choose one of the supported Regions listed for Systems Manager in Systems Manager service endpoints in the Amazon Web Services General Reference. If necessary, first verify with an administrator for your AWS account which Region you should choose.

For more information about access keys, see Managing Access Keys for IAM Users in the IAM User Guide

3. To verify the installation or upgrade, run the following command from the AWS CLI:

   ```shell
   aws ssm help
   ```

   If successful, this command displays a list of available Systems Manager commands.

AWS Tools for PowerShell

**To install or upgrade and then configure the AWS Tools for Windows PowerShell**


   **Tip**
   AWS Tools for PowerShell is frequently updated with new functionality. Upgrade (reinstall) the AWS Tools for PowerShell periodically to ensure that you have access to all the latest functionality.


In this step, you specify credentials that an AWS administrator in your organization has given you, using the following command.

```powershell
Set-AWSCredential -AccessKey AKIAIOSFODNN7EXAMPLE -SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```
Important
When you configure AWS Tools for PowerShell, you can run `Set-DefaultAWSRegion` to specify an AWS Region. Choose one of the supported Regions listed for Systems Manager in Systems Manager service endpoints in the Amazon Web Services General Reference. If necessary, first verify with an administrator for your AWS account which Region you should choose.

For more information about access keys, see Managing Access Keys for IAM Users in the IAM User Guide.

3. To verify the installation or upgrade, run the following command from AWS Tools for PowerShell.

```powershell
Get-AWSCmdletName -Service SSM
```

If successful, this command displays a list of available Systems Manager cmdlets.

### Step 2: Practice installing or updating SSM Agent on an instance

AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure these resources. The agent processes requests from the Systems Manager service in the AWS Cloud, and then runs them as specified in the request. SSM Agent then sends status and execution information back to the Systems Manager service by using the Amazon Message Delivery Service (service prefix: `ec2messages`).

In this task, you install or update the SSM Agent on an EC2 instance.

**Note**
If you are working with your own on-premises servers or VMs, see the following topics:

- Install SSM Agent for a hybrid environment (Windows) (p. 55)
- Install SSM Agent for a hybrid environment (Linux) (p. 52)

**Prerequisites**

An instance profile for Systems Manager must already be attached to the EC2 instance that you update. Refer to the following topics as needed to meet this requirement:

- Create an EC2 instance profile for Systems Manager: Create an IAM instance profile for Systems Manager (p. 30)
- Attach the instance profile to an EC2 instance when you create the instance: Attach an IAM instance profile to an EC2 instance (p. 35)
- Attach the instance profile to an existing EC2 instance: Attaching an IAM Role to an Instance in the Amazon EC2 User Guide

**Windows Server instance**

To practice installing or updating SSM Agent on an EC2 instance for Windows Server, follow the steps in Install and configure SSM Agent on EC2 instances for Windows Server (p. 65).
Linux instance

To practice installing or updating SSM Agent on an EC2 instance for Linux, follow the steps for your Linux operating system type in Manually install SSM Agent on EC2 instances for Linux (p. 69).

Step 3: Try Systems Manager tutorials and walkthroughs

This topic guides you to tutorials, walkthroughs, and basic tasks to help you learn how to use Systems Manager.

Because Systems Manager is a collection of several capabilities, no single walkthrough or tutorial can introduce the entire service. Therefore, we’ve provided links to resources according to the capability for which they provide practice.

In most cases, you do not need to complete additional setup or configuration tasks before you start. You can complete all of the tasks if you have necessary permissions and, where needed, access to one or more managed instances.

In some cases, additional configuration, setup, or experience with Systems Manager are required before you try a tutorial or walkthrough. We have identify those tutorials and walkthroughs as "Advanced".

Compliance

The AWS Systems Manager Configuration Compliance (p. 696) capability scans your fleet of managed instances for patch compliance and configuration inconsistencies.

- Configuration Compliance walkthrough (AWS CLI) (p. 703)

Run Command

The AWS Systems Manager Run Command (p. 829) capability provides you safe, secure remote management of your instances at scale without logging into your servers, replacing the need for bastion hosts, SSH, or remote PowerShell. It provides a simple way of automating common administrative tasks across groups of instances such as registry edits, user management, and software and patch installations.

- Walkthrough: Use the AWS CLI with Run Command (p. 850)
- Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command (p. 853)

Session Manager

The AWS Systems Manager Session Manager (p. 770) capability lets you manage your EC2 instances through an interactive one-click browser-based shell or through the AWS CLI without the need to open inbound ports, maintain bastion hosts, or manage SSH keys.

- Working with Session Manager (p. 812)

Distributor

The AWS Systems Manager Distributor (p. 999) capability lets you package your own software—or find AWS-provided agent software packages, such as AmazonCloudWatchAgent—to install on Systems Manager managed instances.

- Create a package (p. 1005)
• Step 4: Add a package to Distributor (p. 1013)

Patch Manager

The AWS Systems Manager Patch Manager (p. 911) capability helps you select and deploy operating system and software patches automatically across large groups of EC2 instances or on-premises servers and VMs.

• Create a custom patch baseline (p. 959)
• Create a patch group (p. 965)
• Walkthrough: Patch a server environment (command line) (p. 991)

Maintenance Windows

The AWS Systems Manager Maintenance Windows (p. 620) capability lets you define a schedule for performing potentially disruptive actions on your managed instances, such as patching an operating system, updating drivers, or installing software or patches.

• Tutorial: Create and configure a maintenance window (AWS CLI) (p. 642)
• Tutorial: Update a maintenance window (AWS CLI) (p. 676)
• Tutorial: View information about maintenance windows (AWS CLI) (p. 663)
• Tutorial: View information about tasks and task executions (AWS CLI) (p. 674)

State Manager

The AWS Systems Manager State Manager (p. 864) capability helps you maintain consistent configuration of your EC2 instances or on-premises servers and VMs, in a state that you define. Using State Manager, you can control configuration details such as server configurations, anti-virus definitions, firewall settings, and more.

• Creating associations that run MOF files (p. 887)
• Automatically update SSM Agent (CLI) (p. 908)
• Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server (console) (p. 910)

Documents

The AWS Systems Manager documents (p. 1031) capability lets you create and manage SSM documents. An SSM document defines the actions that Systems Manager performs on your managed instances. Systems Manager includes more than a dozen pre-configured documents that you can use by specifying parameters at runtime. Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify.

• Create an SSM document (console) (p. 1086)
• Create an SSM document (command line) (p. 1087)

Parameter Store

The AWS Systems Manager Parameter Store (p. 198) capability provides a centralized store to manage your configuration data, whether plain-text data such as database strings or secrets such as passwords. This allows you to separate your secrets and configuration data from your code. Parameters can be tagged and organized into hierarchies, helping you manage parameters more easily.

• Walkthrough: Create and test a String parameter (console) (p. 267)
• Walkthrough: Create and update a String parameter (AWS CLI) (p. 267)
• Walkthrough: Manage parameters using hierarchies (AWS CLI) (p. 274)
• Advanced: Walkthrough: Create a SecureString parameter and join an instance to a Domain (PowerShell) (p. 272)

Inventory

The AWS Systems Manager Inventory (p. 704) capability collects information about your instances and the software installed on them, helping you to understand your system configurations and installed applications.

• Advanced: Walkthrough: Assign custom inventory metadata to an instance (p. 748)
• Advanced: Walkthrough: Configure your managed instances for Inventory by using the CLI (p. 749)
• Advanced: Walkthrough: Use Resource Data Sync to aggregate inventory data (p. 751)

Automation

The AWS Systems Manager Automation (p. 278) capability allows you to safely automate operations and management tasks across AWS resources. You can automate common IT tasks, safely perform disruptive tasks in bulk, simplify complex tasks, enhance operations security, and use stored configuration scripts share best practices with the rest of your organization.

• Advanced: Walkthrough: Patch a Linux AMI (console) (p. 569)
• Advanced: Walkthrough: Patch a Linux AMI (AWS CLI) (p. 572)
• Advanced: Walkthrough: Patch a Windows Server AMI (p. 576)
• Advanced: Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store (p. 580)
• Advanced: Walkthrough: Patch an AMI and update an Auto Scaling group (p. 586)
• Advanced: Walkthrough: Run the EC2Rescue tool on unreachable instances (p. 591)
• Advanced: Walkthrough: Reset passwords and SSH keys on EC2 instances (p. 596)
• Advanced: Walkthrough: Using Automation with Jenkins (p. 601)
Working with SSM Agent

AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure these resources. The agent processes requests from the Systems Manager service in the AWS Cloud, and then runs them as specified in the request. SSM Agent then sends status and execution information back to the Systems Manager service by using the Amazon Message Delivery Service (service prefix: ec2messages).

If you monitor traffic, you will see your EC2 instances, and any on-premises servers or VMs in your hybrid environment, communicating with ec2messages.* endpoints. For more information, see Reference: ec2messages, ssmmessages, and other API calls (p. 1195). For information about porting SSM Agent logs to Amazon CloudWatch Logs, see Monitoring AWS Systems Manager (p. 1134).

Keeping SSM Agent up-to-date

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Note
AMIs that include SSM Agent by default can take up to two weeks to be updated with the newest version of SSM Agent. We recommend that you configure even more frequent automated updates to SSM Agent.
Updated versions of SSM Agent are rolled out to new AWS Regions at different times. For this reason, you might receive the "Unsupported on current platform" or "updating amazon-ssm-agent to an older version, please enable allow downgrade to proceed" error when trying to deploy a new version of SSM Agent in a Region.

SSM Agent and the Instance Metadata Service (IMDS)

Systems Manager relies on EC2 instance metadata to function correctly. Systems Manager can access instance metadata using either version 1 or version 2 of the Instance Metadata Service (IMDSv1 and IMDSv2). For more information, see Instance Metadata and User Data in the Amazon EC2 User Guide for Linux Instances.

SSM Agent credentials precedence

The SSM Agent requires permissions provided by an IAM role to communicate with Systems Manager. It's important to understand how these credentials are sourced and evaluated by the SSM Agent. Otherwise, previously configured credentials on your managed instances might supersede your desired credential provider. SSM Agent credentials are evaluated in the following order.

1. Environment variables ($HOME, %USERPROFILE%)
2. Shared credentials file ($HOME/.aws/credentials, %USERPROFILE%\aws\credentials)
3. Instance profile

About the local ssm-user account
Starting with version 2.3.50.0 of SSM Agent, the agent creates a local user account called `ssm-user` and adds it to `/etc/sudoers` (Linux) or to the Administrators group (Windows). On agent versions before 2.3.612.0, the account is created the first time SSM Agent starts or restarts after installation. On version 2.3.612.0 and later, the `ssm-user` account is created the first time a session is started on an instance. This `ssm-user` is the default OS user when a Session Manager session is started. You can change the permissions by moving `ssm-user` to a less-privileged group or by changing the `sudoers` file. The `ssm-user` account is not removed from the system when SSM Agent is uninstalled.

On Windows Server, SSM Agent handles setting a new password for the `ssm-user` account when each session starts. No passwords are set for `ssm-user` on Linux managed instances.

Starting with SSM Agent version 2.3.612.0, the `ssm-user` account is not created automatically on Windows Server machines that are being used as domain controllers. To use Session Manager on a Windows Server domain controller, you must create the `ssm-user` account manually if it isn’t already present.

**Important**

In order for the ssm-user account to be created, the instance profile attached to the instance must provide the necessary permissions. For information, see [Verify or create an IAM instance profile with Session Manager permissions](p. 776).

### AMIs with SSM Agent preinstalled

SSM Agent is preinstalled, by default, on the following Amazon Machine Images (AMIs):

- Windows Server 2008-2012 R2 AMIs published in November 2016 or later
- Windows Server 2016 and 2019
- Amazon Linux
- Amazon Linux 2
- Ubuntu Server 16.04
- Ubuntu Server 18.04
- Amazon ECS-Optimized

You must manually install SSM Agent on EC2 instances created from other Linux AMIs. You must also manually install SSM Agent on on-premises servers or VMs in your hybrid environment. For more information, see [Setting up AWS Systems Manager for hybrid environments](p. 43).

### SSM Agent on GitHub

The source code for SSM Agent is available on [GitHub](https://github.com) so that you can adapt the agent to meet your needs. We encourage you to submit pull requests for changes that you would like to have included. However, Amazon Web Services does not currently provide support for running modified copies of this software.

### Contents

- Installing and configuring SSM Agent on Windows Server instances (p. 65)
- Installing and configuring SSM Agent on EC2 instances for Linux (p. 69)
- Getting the currently installed SSM Agent version (p. 89)
- View SSM Agent logs (p. 90)
- Restrict access to root-level commands through SSM Agent (p. 92)
- Automate updates to SSM Agent (p. 93)
- Subscribe to SSM Agent notifications (p. 94)
Installing and configuring SSM Agent on Windows Server instances

SSM Agent is installed by default on instances created from Windows Server 2016 and Windows Server 2019 Amazon Machine Images (AMIs), and on instances created from Windows Server 2008-2012 R2 AMIs published in November 2016 or later.

Windows Server AMIs published before November 2016 use the EC2Config service to process requests and configure instances.

Unless you have a specific reason for using the EC2Config service, or an earlier version of SSM Agent, to process Systems Manager requests, we recommend that you download and install the latest version of SSM Agent to each of your EC2 instances or hybrid instances that are configured for Systems Manager.

Important
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

To view details about the different versions of SSM Agent, see the release notes.

Topics
- Install and configure SSM Agent on EC2 instances for Windows Server (p. 65)
- Configure SSM Agent to use a proxy for Windows Server instances (p. 67)

Install and configure SSM Agent on EC2 instances for Windows Server

SSM Agent is installed by default on instances created from Windows Server 2016 and Windows Server 2019 Amazon Machine Images (AMIs), and on instances created from Windows Server 2008-2012 R2 AMIs published in November 2016 or later.

If your instance is a Windows Server 2008-2012 R2 instance created before November 2016, then EC2Config processes Systems Manager requests on your instance. We recommend that you upgrade your existing instances to use the latest version of EC2Config. By using the latest EC2Config installer, you install SSM Agent side-by-side with EC2Config. This side-by-side version of SSM Agent is compatible with your instances created from earlier Windows Server AMIs and enables you to use SSM features published after November 2016. For information about how to install the latest version of the EC2Config service, see Installing the Latest Version of EC2Config in the Amazon EC2 User Guide for Windows Instances.

Important
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.
If necessary, you can manually download and install the latest version of SSM Agent on your EC2 instance for Windows Server by using the following procedure.

**Important**
This procedure applies to installing or reinstalling SSM Agent on an EC2 instance for Windows Server. If you need to install the agent on an on-premises server or a virtual machine (VM) so it can be used with Systems Manager, see Install SSM Agent for a hybrid environment (Windows) (p. 55).

**To manually download and install the latest version of SSM Agent**

1. Log in to your instance by using Remote Desktop or Windows PowerShell.
2. Download the latest version of SSM Agent to your instance. You can download using either PowerShell commands or a direct download link.

**Note**
The URLs in this step let you download SSM Agent from any AWS Region. If you want to download the agent from a specific Region, use a Region-specific URL instead:

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

`region` represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported `region` values, see the `Region` column in Systems Manager service endpoints in the Amazon Web Services General Reference.

**PowerShell**

Run the following three PowerShell commands in order. These commands enable you to download SSM Agent without adjusting Internet Explorer (IE) Enhanced Security settings, and then install the agent and remove the installation file.

**64-bit**

```
Invoke-WebRequest -Uri "https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe" -OutFile $env:USERPROFILE\Desktop\SSMAgent_latest.exe
```

**32-bit**

```
Invoke-WebRequest -Uri "https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_386/AmazonSSMAgentSetup.exe" -OutFile $env:USERPROFILE\Desktop\SSMAgent_latest.exe
```

```
Start-Process -FilePath $env:USERPROFILE\Desktop\SSMAgent_latest.exe -ArgumentList "/S"
```

```
rm -Force $env:USERPROFILE\Desktop\SSMAgent_latest.exe
```

**Direct download**

Download the latest version of SSM Agent to your instance by using the following link. If you want, update this URL with an AWS Region-specific URL.

```
https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/windows_amd64/AmazonSSMAgentSetup.exe
```
Configure SSM Agent to use a proxy for Windows Server instances

The information in this topic applies to Windows Server instances created in or after November 2016 that do not use the Nano installation option.

Important
SSM Agent requires Windows PowerShell 3.0 or later to run certain SSM Documents on Windows Server instances (for example, the legacy AWS-ApplyPatchBaseline document). Verify that your Windows Server instances are running Windows Management Framework 3.0 or later. This framework includes Windows PowerShell. For more information, see Windows Management Framework 3.0.

Configure SSM Agent to use a proxy for Windows Server instances

If your instance is a Windows Server 2008-2012 R2 instance created before November 2016, then EC2Config processes Systems Manager requests on your instance. We recommend that you upgrade your existing instances to use the latest version of EC2Config. By using the latest EC2Config installer, you install SSM Agent side-by-side with EC2Config. This side-by-side version of SSM Agent is compatible with your instances created from earlier Windows AMIs and enables you to use SSM features published after November 2016. For information about how to install the latest version of the EC2Config service, see Installing the Latest Version of EC2Config in the Amazon EC2 User Guide for Windows Instances. If you do not upgrade to the latest version of EC2Config and use EC2Config to process Systems Manager requests, you must configure proxy settings for EC2Config. For information about configuring EC2Config to use a proxy, see Configure Proxy Settings for the EC2Config Service.

To configure SSM Agent to use a proxy

1. Using Remote Desktop or Windows PowerShell, connect to the instance that you would like to configure to use a proxy.

2. Run the following command block in PowerShell. Replace hostname and port with the information about your proxy:

   ```powershell
   $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
   ```

   After running the preceding command, you can review the SSM Agent logs to confirm the proxy settings were applied. Entries in the logs look similar to the following. For more information about SSM Agent logs, see View SSM Agent logs (p. 90).
To reset SSM Agent proxy configuration

1. Using Remote Desktop or Windows PowerShell, connect to the instance to configure.
2. If you connected using Remote Desktop, launch PowerShell as an administrator.
3. Run the following command block in PowerShell:

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

SSM Agent proxy setting precedence

When configuring proxy settings for the SSM Agent on Windows Server instances, it's important to understand these settings are evaluated and applied to the agent configuration when the SSM Agent is started. How you configure your proxy settings for a Windows Server instance can determine whether other settings might supersede your desired settings. SSM Agent proxy settings are evaluated in the following order.

1. AmazonSSMAgent Registry settings (HKLM:\SYSTEM\CurrentControlSet\Services\AmazonSSMAgent)
2. System environment variables (http_proxy, https_proxy, no_proxy)
3. LocalSystem user account environment variables (http_proxy, https_proxy, no_proxy)
4. WinHTTP proxy settings
5. Internet Explorer settings

SSM Agent proxy settings and Systems Manager services

If you configured the SSM Agent to use a proxy and are using AWS Systems Manager services, such as Run Command and Patch Manager, that use PowerShell or the Windows Update client during their execution on Windows Server instances, you must configure additional proxy settings. Otherwise, the operation might fail because proxy settings used by PowerShell and the Windows Update client are not inherited from the SSM Agent proxy configuration.

For Run Command, you must configure WinINet proxy settings on your Windows Server instances. The following PowerShell commands return the current WinINet proxy settings, and apply your proxy settings to WinINet.

```powershell
[System.Net.WebRequest]::DefaultWebProxy
#proxyServer = "http://hostname:port"
#proxyBypass = "169.254.169.**"
```

For Patch Manager, you must configure system-wide proxy settings so the Windows Update client can scan for and download updates. We recommend that you use Run Command to run the following commands because they run on the SYSTEM account, and the settings apply system-wide. The following `netsh` commands return the current proxy settings, and apply your proxy settings to the local system.

```
netsh winhttp show proxy
netsh winhttp set proxy hostname:port
```

For more information about using Run Command, see Running commands using Systems Manager Run Command (p. 833).

## Installing and configuring SSM Agent on EC2 instances for Linux

SSM Agent processes Systems Manager requests and configures your machine as specified in the request. Use the following procedures to install, configure, or uninstall SSM Agent.

### Important
SSM Agent is preinstalled, by default, on the following Amazon Machine Images (AMIs):

- Amazon Linux
- Amazon Linux 2
- Ubuntu Server 16.04
- Ubuntu Server 18.04
- Amazon ECS-Optimized

You must manually install SSM Agent on EC2 instances created from other Linux AMIs.

The source code for SSM Agent is available on GitHub so that you can adapt the agent to meet your needs. We encourage you to submit pull requests for changes that you would like to have included. However, AWS does not currently provide support for running modified copies of this software.

### Note
To view details about the different versions of SSM Agent, see the release notes.

### Topics
- Manually install SSM Agent on EC2 instances for Linux (p. 69)
- Configure SSM Agent to use a proxy (p. 86)
- Uninstall SSM Agent from Linux instances (p. 89)

## Manually install SSM Agent on EC2 instances for Linux

After you manually install SSM Agent, you can automatically update SSM Agent on your instances when new versions become available by using Systems Manager State Manager. For more information, see Automatically update SSM Agent (CLI) (p. 908).
Important
These procedures apply to installing or reinstalling SSM Agent on EC2 instances for Linux. If you need to install the agent on an on-premises server or a virtual machine (VM) so it can be used with Systems Manager, see Install SSM Agent for a hybrid environment (Linux) (p. 52).

Download SSM Agent from a specific AWS Region

The URLs in the scripts in the topics later this section let you download SSM Agent from any AWS Region. If you want to download the agent from a specific Region, copy the URL for your operating system, and then replace `region` with an appropriate value.

`region` represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported `region` values, see the `Region` column in Systems Manager service endpoints in the Amazon Web Services General Reference.

For example, to download SSM Agent for Amazon Linux, RHEL, CentOS, and SLES 64-bit from the US West 1 Region, use the following URL:

```
```

- Amazon Linux, RHEL, CentOS, and SLES 64-bit:
  ```
  ```
- Amazon Linux, RHEL, and CentOS 32-bit:
  ```
  ```
- Ubuntu Server 64-bit:
  ```
  ```
- Ubuntu Server 32-bit:
  ```
  ```
- Raspbian:
  ```
  ```

Important
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Operating systems
- Amazon Linux (p. 71)
- Amazon Linux 2 (p. 72)
- CentOS (p. 73)
- Debian Server (p. 75)
- Oracle Linux (p. 76)
- Raspbian (p. 77)
Manually install SSM Agent on Amazon Linux instances

Connect to your Amazon Linux instance and perform the following steps to install SSM Agent.

**Note**
If you use a `yum` command to update SSM Agent on a managed instance after the agent has been installed or updated using the SSM document `AWS-UpdateSSMAgent`, you might see the following message: "Warning: RPMDB altered outside of yum." This message is expected and can be safely ignored.

Perform these steps on each instance that will run commands using Systems Manager.

**Note**
SSM Agent is installed, by default, on Amazon Linux base AMIs dated 2017.09 and later. SSM Agent is also installed, by default, on Amazon Linux 2 AMIs. You must manually install SSM Agent on other versions of Linux.

Instances created from an Amazon Linux AMI that are using a proxy must be running a current version of the Python `requests` module in order to support Patch Manager operations. For more information, see Upgrade the Python requests module on Amazon Linux instances that use a proxy server (p. 88).

**To install SSM Agent on Amazon Linux**

1. Use one of the following commands to download and run the SSM Agent installer.
   
   **Note**
   Even though the following download URLs show 'ec2-downloads-windows', these are the correct URLs for downloading Amazon Linux.

   **Intel (x86_64) 64-bit instances:**
   ```bash
   sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
   ```

   **ARM (arm64) 64-bit instances:**
   ```bash
   sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm
   ```

   **Intel (x86) 32-bit instances:**
   ```bash
   ```

2. Run the following command to determine if SSM Agent is running. The command should return the message "amazon-ssm-agent is running."

   Check the status of the agent.
   ```bash
   sudo status amazon-ssm-agent
   ```

3. Run the following commands if the previous command returns the message "amazon-ssm-agent is stopped."

   ```bash
   ```
a. Start the service.

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

b. Check the status of the agent.

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

**Important**
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

**Manually install SSM Agent on Amazon Linux 2 instances**

Connect to your Amazon Linux 2 instance and perform the following steps to install SSM Agent.

**Note**
If you use a `yum` command to update SSM Agent on a managed instance after the agent has been installed or updated using the SSM document `AWS-UpdateSSMAgent`, you might see the following message: "Warning: RPMDB altered outside of yum." This message is expected and can be safely ignored.

Perform these steps on each instance that will run commands using Systems Manager.

**Important**
SSM Agent is installed, by default, on Amazon Linux base AMIs dated 2017.09 and later. SSM Agent is also installed, by default, on Amazon Linux 2 AMIs. You must manually install SSM Agent on other versions of Linux.

**To install SSM Agent on Amazon Linux 2**

1. Use one of the following commands to download and run the SSM Agent installer.

**Note**
Even though the following download URLs show 'ec2-downloads-windows', these are the correct URLs for downloading Amazon Linux and Amazon Linux 2.

**Intel (x86_64) 64-bit instances:**

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

**ARM (arm64) 64-bit instances:**

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm
```

**Intel (x86) 32-bit instances:**

```
```
2. Run the following command to determine if SSM Agent is running. The command should return the message "amazon-ssm-agent is running."

Check the status of the agent.

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

3. Run the following commands if the previous command returns the message "amazon-ssm-agent is stopped."

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

   b. Check the status of the agent.

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

**Important**

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

### Manually install SSM Agent on CentOS instances

Connect to your CentOS instance and perform the following steps to install the SSM Agent.

**Note**

If you use a yum command to update SSM Agent on a managed instance after the agent has been installed or updated using the SSM document AWS-UpdateSSMAgent, you might see the following message: "Warning: RPMDB altered outside of yum." This message is expected and can be safely ignored.

Perform these steps on each instance that will run commands using Systems Manager.

**CentOS 7.x**

#### To install SSM Agent on CentOS 7.x

1. Use one of the following commands to download and run the SSM Agent installer.

   **Note**
   
   Even though the following download URLs show 'ec2-downloads-windows', these are the correct URLs.
   
   64-bit instances:
   
   ```
   sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
   ```
   
   32-bit instances:

2. Run one of the following commands to determine if SSM Agent is running. The command should return the message `amazon-ssm-agent is running`.

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

3. Run the following commands if the previous command returned `amazon-ssm-agent is stopped`.
   a. Start the service.
      ```bash
      sudo systemctl enable amazon-ssm-agent
      sudo systemctl start amazon-ssm-agent
      ```
   b. Check the status of the agent.
      ```bash
      sudo systemctl status amazon-ssm-agent
      ```

**CentOS 6.x**

**To install SSM Agent on CentOS 6.x**

1. Use one of the following commands to download and run the SSM Agent installer.

   **Note**
   Even though the following download URLs show `ec2-downloads-windows`, these are the correct URLs.

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

   **32-bit instances:**
   ```bash
   ```

2. Run one of the following commands to determine if SSM Agent is running. The command should return the message `amazon-ssm-agent is running`.

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

3. Run the following commands if the previous command returned `amazon-ssm-agent is stopped`.
   a. Start the service.
      ```bash
      sudo start amazon-ssm-agent
      ```
   b. Check the status of the agent.
Important
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Manually install SSM Agent on Debian Server instances
Connect to your Debian Server instance and perform the steps in one of following procedures to install SSM Agent on each instance that will run commands using Systems Manager.

Install SSM Agent on Debian Server instances
Debian Server 9 64-bit (deb)

To install SSM Agent on Debian Server 9 64-bit instances (with deb installer package)

1. Connect to your Debian Server instance and perform the following steps to install SSM Agent. Perform these steps on each instance that will run commands using Systems Manager.

Create a temporary directory on the instance.

```bash
mkdir /tmp/ssm
```

Change to the temporary directory.

```bash
cd /tmp/ssm
```

Run the following commands.

**Note**
Even though the following download URL shows 'ec2-downloads-windows', this is the correct URL.

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

```bash
sudo dpkg -i amazon-ssm-agent.deb
```

2. Run following command to determine if SSM Agent is running.

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

3. Run the following command to start the service if the previous command returned `amazon-ssm-agent` is stopped, inactive, or disabled.

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

4. Run the following command to check the status of the agent.
Debian Server 8 64-bit (deb)

To install SSM Agent on Debian Server 8 64-bit instances (with deb installer package)

1. Connect to your Debian Server instance and perform the following steps to install SSM Agent. Perform these steps on each instance that will run commands using Systems Manager.

   Create a temporary directory on the instance.

   ```bash
   mkdir /tmp/ssm
   ```

   Change to the temporary directory.

   ```bash
   cd /tmp/ssm
   ```

   Run the following commands.

   **Note**
   Even though the following download URL shows 'ec2-downloads-windows', this is the correct URL.

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

   ```bash
   sudo dpkg -i amazon-ssm-agent.deb
   ```

2. Run following command to determine if SSM Agent is running.

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

3. Run the following command to start the service if the previous command returned amazon-ssm-agent is stopped, inactive, or disabled.

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

4. Run the following command to check the status of the agent.

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

**Important**
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Manually install SSM Agent on Oracle Linux instances

Connect to your Oracle Linux 7.5 or 7.7 instance and perform the following steps to install SSM Agent.
Note
If you use a `yum` command to update SSM Agent on a managed instance after the agent has been installed or updated using the SSM document `AWS-UpdateSSMAgent`, you might see the following message: "Warning: RPMDB altered outside of yum." This message is expected and can be safely ignored.

Perform these steps on each instance that will run commands using Systems Manager.

To install SSM Agent on Oracle Linux 7.5 and 7.7

1. Use the following command to download and run the SSM Agent installer.
   
   **Note**
   Even though the following download URLs show 'ec2-downloads-windows', these are the correct URLs.
   
   Intel (x86_64) 64-bit instances:
   
   ```bash
   sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
   ```

2. Run the following command to determine if SSM Agent is running. The command should return the message `amazon-ssm-agent is running`.
   
   ```bash
   sudo systemctl status amazon-ssm-agent
   ```

3. Run the following commands if the previous command returned `amazon-ssm-agent is stopped`.
   
   a. Start the service.
   
   ```bash
   sudo systemctl enable amazon-ssm-agent
   sudo systemctl start amazon-ssm-agent
   ```

   b. Check the status of the agent.
   
   ```bash
   sudo systemctl status amazon-ssm-agent
   ```

Important
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Manually install SSM Agent on Raspbian instances

This section includes information about how to install SSM Agent on Raspbian Jessie and Raspbian Stretch, including Raspberry Pi (32-bit) devices.

Before You Begin

To set up your Raspbian devices as Systems Manager managed instances, you need to create a managed-instance activation. After you complete the activation, you receive an activation code and ID. This code/
ID combination functions like an Amazon EC2 access ID and secret key to provide secure access to the Systems Manager service from your managed instances. Store the activation code and ID in a safe place. For more information about the activation process, see Setting up AWS Systems Manager for hybrid environments (p. 43).

Connect to your Raspbian device and perform the following steps to install the SSM Agent. Perform these steps on each instance that will run commands using Systems Manager.

**To install SSM Agent on Raspbian devices**

1. Create a temporary directory on the instance.
   ```
   mkdir /tmp/ssm
   ```

2. Use the following command to download and run the SSM Agent installer.
   ```
   ```

3. Run the following command to install SSM Agent.
   ```
   sudo dpkg -i /tmp/ssm/amazon-ssm-agent.deb
   ```

4. Run the following command to stop SSM Agent.
   ```
   sudo service amazon-ssm-agent stop
   ```

5. Run the following command to register the agent using the managed-instance activation code and ID you received when you completed the managed-instance activation process.
   ```
   sudo amazon-ssm-agent -register -code "code" -id "ID" -region "region"
   ```

6. Run the following command to start SSM Agent.
   ```
   sudo service amazon-ssm-agent start
   ```

**Note**

- If you see the following error in the SSM Agent error logs, then the machine ID did not persist after a reboot:
  
  Unable to load instance associations, unable to retrieve associations unable to retrieve associations error occurred in RequestManagedInstanceRoleToken: MachineFingerprintDoesNotMatch: Fingerprint does not match
  
  Run the following command to make the machine ID persist after a reboot.
  ```
  umount /etc/machine-id
  systemctl-machine-id-setup
  ```

- An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.
Manually install SSM Agent on Red Hat Enterprise Linux instances

Connect to your RHEL instance and perform the following steps to install SSM Agent.

**Note**
If you use a `yum` command to update SSM Agent on a managed instance after the agent has been installed or updated using the SSM document `AWS-UpdateSSMAgent`, you might see the following message: "Warning: RPMDB altered outside of yum." This message is expected and can be safely ignored.

Perform these steps on each instance that will run commands using Systems Manager.

RHEL 8.x

**To install SSM Agent on Red Hat Enterprise Linux 8.x**

1. Ensure that either Python 2 or Python 3 is installed on your RHEL 8 instance. This is required in order for SSM Agent to work properly.
2. Use one of the following commands to download and run the SSM Agent installer.
   
   **Note**
   Even though the following download URLs show 'ec2-downloads-windows', these are the correct URLs.

   **Intel (x86_64) 64-bit instances:**
   ```
sudo dnf install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm
   ```

   **ARM (arm64) 64-bit instances:**
   ```
sudo dnf install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm
   ```

   **Intel (x86) 32-bit instances:**
   ```
   ```

3. Run one of the following commands to determine if SSM Agent is running. The command should return the message `amazon-ssm-agent is running`.
   ```
sudo systemctl status amazon-ssm-agent
   ```

4. Run the following commands if the previous command returned `amazon-ssm-agent is stopped`.
   a. Start the service.
   ```
sudo systemctl enable amazon-ssm-agent
   ```
   ```
sudo systemctl start amazon-ssm-agent
   ```
   b. Check the status of the agent.
   ```
sudo systemctl status amazon-ssm-agent
   ```
RHEL 7.x

**To install SSM Agent on Red Hat Enterprise Linux 7.x**

1. Use one of the following commands to download and run the SSM Agent installer.

   **Note**
   Even though the following download URLs show 'ec2-downloads-windows', these are the correct URLs.

   Intel (x86_64) 64-bit instances:

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

   ARM (arm64) 64-bit instances:

   ```
   sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm
   ```

   Intel (x86) 32-bit instances:

   ```
   ```

2. Run one of the following commands to determine if SSM Agent is running. The command should return the message **amazon-ssm-agent is running**.

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

3. Run the following commands if the previous command returned **amazon-ssm-agent is stopped**.

   a. Start the service.

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

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

   b. Check the status of the agent.

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

RHEL 6.x

**To install SSM Agent on Red Hat Enterprise Linux 6.x**

1. Use one of the following commands to download and run the SSM Agent installer.
Note
Even though the following download URLs show ‘ec2-downloads-windows’, these are the correct URLs.

Intel (x86_64) 64-bit instances:

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

ARM (arm64) 64-bit instances:

```
sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_arm64/amazon-ssm-agent.rpm
```

Intel (x86) 32-bit instances:

```
```

2. Run one of the following commands to determine if SSM Agent is running. The command should return the message _amazon-ssm-agent is running_.

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

3. Run the following commands if the previous command returned _amazon-ssm-agent is stopped_.

   a. Start the service.

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

   b. Check the status of the agent.

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

Important
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Manually install SSM Agent on SUSE Linux Enterprise Server 12 instances

Connect to your SLES instance and perform the following steps to install the SSM Agent. Perform these steps on each instance that will run commands using Systems Manager.

To install SSM Agent on SUSE Linux Enterprise Server

1. Create a temporary directory on the instance.
mkdir /tmp/ssm

2. Change to the temporary directory.

cd /tmp/ssm

3. Run the following commands one at a time to download and run the SSM Agent installer.

   Note
   Even though the following download URL shows ‘ec2-downloads-windows’, this is the correct URL.

   64-bit instances:

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

   sudo rpm --install amazon-ssm-agent.rpm

4. Run the following command to determine if SSM Agent is running. The command should return the message `amazon-ssm-agent is running`.

   sudo systemctl status amazon-ssm-agent

5. Run the following commands if the previous command returns the message `amazon-ssm-agent is stopped`.

   a. Start the service.

      sudo systemctl enable amazon-ssm-agent

      sudo systemctl start amazon-ssm-agent

   b. Check the status of the agent.

      sudo systemctl status amazon-ssm-agent

**Important**
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

**Manually install SSM Agent on Ubuntu Server instances**

Connect to your Ubuntu Server instance and perform the steps in one of following procedures to install SSM Agent on each instance that will run commands using Systems Manager.

**About SSM Agent installations on 64-bit Ubuntu Server 16.04 instances**

Beginning with instances created from Ubuntu Server 16.04 AMIs identified with 20180627, SSM Agent is pre-installed using Snap packages. For example: ubuntu/images/hvm-ssd/ubuntu-
On instances created from earlier AMIs, you should continue using deb installer packages.

**Important**
Be aware that if an instance has more than one installation of the SSM Agent (for example, one installed using a Snap, one installed using a deb installer), your agent operations will not work correctly.

You can check the source AMI ID for an instance following these steps:

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. In the left navigation, choose **Instances**.
3. Select an instance.
4. On the **Description** tab, locate the value in the **AMI ID** field.

For instances created from a 64-bit Ubuntu Server 16.04 AMI, be sure to follow the correct procedure for your SSM Agent installation type:

- **Instances created from AMIs with identifier 20180627 or later**: Install SSM Agent on Ubuntu Server instances (p. 83)
- **Instances created from AMIs earlier than 20180627**: Install SSM Agent on Ubuntu Server instances (p. 83)

**Install SSM Agent on Ubuntu Server instances**

**Ubuntu Server 18.04 and 16.04 LTS 64-bit (Snap)**

**To install SSM Agent on Ubuntu Server 18.04 and 16.04 LTS 64-bit instances (with Snap package)**

1. SSM Agent is installed, by default, on Ubuntu Server 18.04 and on 16.04 LTS 64-bit AMIs with an identifier of 20180627 or later. For more information about version 16.04 AMIs, see [Manually install SSM Agent on Ubuntu Server instances (p. 82)](#).

   You can use the following script if you need to install SSM Agent on an on-premises server or if you need to reinstall the agent. You don't need to specify a URL for the download, because the `snap` command automatically downloads the agent from the Snap app store at [https://snapcraft.io](https://snapcraft.io).

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

   **Note**
   Note the following details about SSM Agent on Ubuntu Server 18.04 and 16.04:

   - Because of a known issue with Snap, you might see a **Maximum timeout exceeded** error with `snap` commands. If you get this error, run the following commands one at a time to start the agent, stop it, and check its status:

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

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

     ```bash
     sudo systemctl status snap.amazon-ssm-agent.amazon-ssm-agent.service
     ```
• On Ubuntu Server 18.04 and 16.04, SSM Agent installer files, including agent binaries and config files, are stored in the following directory: `/snap/amazon-ssm-agent/current/`. If you make changes to any configuration files in this directory, then you must copy these files from the `/snap` folder to the `/etc/amazon/ssm/` folder. Log and library files have not changed (`/var/lib/amazon/ssm`, `/var/log/amazon/ssm`).

• On Ubuntu Server 18.04, use Snaps only. Don't install deb packages. Also verify that only one instance of the agent is installed and running on your instances.

• On Ubuntu Server 18.04 and 16.04, SSM Agent provides support for the arm64 processor architecture.

• On Ubuntu Server 16.04, SSM Agent is installed using either Snaps or deb installation packages, depending on the version of the 16.04 AMI. For more information, see Manually install SSM Agent on Ubuntu Server instances (p. 82).

2. Run the following command to determine if SSM Agent is running.

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

3. Run the following command to start the service if the previous command returned `amazon-ssm-agent is stopped, inactive, or disabled`.

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

4. Check the status of the agent.

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

Ubuntu Server 16.04 and 14.04 64-bit (deb)

**To install SSM Agent on Ubuntu Server 16.04 and 14.04 64-bit instances (with deb installer package)**

1. You can use the following script if you need to install SSM Agent on an on-premises server or if you need to reinstall the agent.

   **Important**
   SSM Agent is installed by default on instances created from Ubuntu Server 16.04 LTS 64-bit AMIs with an identifier of 20180627 or later. Instances created from AMIs with earlier identifiers, for example 20171121.1 and 20180522, should continue to use deb installers.

   If SSM Agent is installed on your instance in conjunction with a Snap and you install or update SSM Agent using a deb installer package, the installation or SSM Agent operations may fail. For more information, see Manually install SSM Agent on Ubuntu Server instances (p. 82).

   Create a temporary directory on the instance.

   ```bash
   mkdir /tmp/ssm
   ```

   Change to the temporary directory.

   ```bash
   cd /tmp/ssm
   ```

   Run the following commands.
**Note**
Even though the following download URL shows 'ec2-downloads-windows', this is the correct URL.

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

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

2. **Run one of the following commands to determine if SSM Agent is running.**

   **Ubuntu Server 16.04:**

   ```
sudo systemctl status amazon-ssm-agent
   
   Ubuntu Server 14.04:
   
   sudo status amazon-ssm-agent
   ```

3. **Run one of the following commands to start the service if the previous command returned amazon-ssm-agent is stopped, inactive, or disabled.**

   **Ubuntu Server 16.04:**

   ```
sudo systemctl enable amazon-ssm-agent
   
   Ubuntu Server 14.04:
   
   sudo start amazon-ssm-agent
   ```

4. **Run one of the following commands to check the status of the agent.**

   **Ubuntu Server 16.04:**

   ```
sudo systemctl status amazon-ssm-agent
   
   Ubuntu Server 14.04:
   
   sudo status amazon-ssm-agent
   ```

**Ubuntu Server 16.04 and 14.04 32-bit**

**To install SSM Agent on Ubuntu Server 16.04 and 14.04 32-bit instances**

1. **Create a temporary directory on the instance.**

   ```
   mkdir /tmp/ssm
   
   Change to the temporary directory.
   
   cd /tmp/ssm
   ```
Run the following commands.

**Note**
Even though the following download URL shows 'ec2-downloads-windows', this is the correct URL.

```
```

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

2. Run the following command to determine if SSM Agent is running:

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

3. Run the following commands if the previous command returned `amazon-ssm-agent` is stopped, inactive, or disabled.

   a. Start the agent:
   
   ```
sudo start amazon-ssm-agent
   ```

   b. Check the status of the agent:
   
   ```
sudo status amazon-ssm-agent
   ```

**Important**
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

## Configure SSM Agent to use a proxy

You can configure SSM Agent to communicate through an HTTP proxy by adding the `http_proxy`, `https_proxy`, and `no_proxy` settings to an `amazon-ssm-agent.override` configuration file. An override file also preserves the proxy settings if you install newer or older versions of SSM Agent. This section includes procedures for `upstart` and `systemd` environments.

**Note**
Instances created from an Amazon Linux AMI that are using a proxy must be running a current version of the Python `requests` module in order to support Patch Manager operations. For more information, see Upgrade the Python requests module on Amazon Linux instances that use a proxy server (p. 88).

**Topics**
- Configure SSM Agent to use a proxy (upstart) (p. 87)
- Configure SSM Agent to use a proxy (systemd) (p. 87)
- Upgrade the Python requests module on Amazon Linux instances that use a proxy server (p. 88)
Configure SSM Agent to use a proxy (upstart)

1. Connect to the instance where you installed SSM Agent.
2. Open a simple editor like VIM, and depending on whether you're using an HTTP proxy server or HTTPS proxy server, specify one of the following setting options.

**HTTP proxy server:**

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

**HTTPS proxy server:**

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

**Note**

You must add the `no_proxy` setting to the file and specify the IP address listed here. It is the instance metadata endpoint for Systems Manager. Without this IP address, calls to Systems Manager fail.

3. Save the file as `amazon-ssm-agent.override` in the following location:

```
/etc/init/
```

4. Stop and restart SSM Agent using the following commands:

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

**Note**

For more information about working with `.override` files in Upstart environments, see `init: Upstart init daemon job configuration`.

Configure SSM Agent to use a proxy (systemd)

The steps in the following procedure describe how to configure SSM Agent to use a proxy in systemd environments. Some of the steps in this procedure contain explicit instructions for Ubuntu Server instances installed by using Snap.

1. Connect to the instance where you installed SSM Agent.
2. Run the following command:

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

For Ubuntu Server instances installed by using a snap, run the following command:

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

3. Open a simple editor like VIM, and depending on whether you're using an HTTP proxy server or HTTPS proxy server, specify one of the following setting options.

**HTTP proxy server:**

```
[Service]
```
Configure SSM Agent to use a proxy

HTTPS proxy server:

Note
You must add the no_proxy setting to the file and specify the IP address listed here. It is the instance metadata endpoint for Systems Manager. Without this IP address, calls to Systems Manager fail.

4. Save your changes. The system creates a file named `amazon-ssm-agent.override` (or `override.conf` on Amazon Linux 2) instances in the `etc/systemd/system/amazon-ssm-agent.service.d` folder.

5. Restart SSM Agent by using the following commands:

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

For Ubuntu Server instances installed by using a snap, restart SSM Agent by using the following command:

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

Note
For more information about working with .override files in systemd environments, see Modifying Existing Unit Files.

Upgrade the Python requests module on Amazon Linux instances that use a proxy server

To patch an instance that is using a proxy and that was created from an Amazon Linux AMI, Patch Manager requires a recent version of the Python requests module to be installed on the instance. We recommend always upgrading to the most recently released version.

To ensure the latest version of the Python requests module is installed, follow these steps:

1. Sign in to the Amazon Linux instance, or use the AWS-RunShellScript SSM document in Run Command, and run the following command on the instance:

```
pip list | grep requests
```

• If the module is installed, the request returns the version number in a response similar to the following:

```
requests (1.2.3)
```

• If the module is not installed, run the following command to install it:
pip install requests

• If pip itself is not installed, run the following command to install it:

   sudo yum install -y python-pip

2. If the module is installed, but the version listed is earlier than 2.18.4 (such as 1.2.3 shown in the previous step), run the following command to upgrade to the latest version of the Python requests module:

   pip install requests --upgrade

Uninstall SSM Agent from Linux instances

Use the following commands to uninstall SSM Agent.

Amazon Linux, Amazon Linux 2, RHEL, and CentOS

   sudo yum erase amazon-ssm-agent -y

Ubuntu Server

• deb package installations:

   sudo dpkg -r amazon-ssm-agent

• snap package installations:

   sudo snap remove amazon-ssm-agent

Debian Server

   sudo dpkg -r amazon-ssm-agent

SLES

   sudo rpm --erase amazon-ssm-agent

Getting the currently installed SSM Agent version

Certain Systems Manager functionalities have prerequisites that include a minimum SSM Agent version be installed on your managed instances. You can get the currently installed SSM Agent version on your managed instances using the AWS Systems Manager console, or by logging in to your instances.

The following procedures describe how to get the currently installed SSM Agent version on your managed instances.

To get the currently installed SSM Agent version using the console

2. In the navigation pane, choose **Managed Instances**.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (Ξ) to open the navigation pane, and then choose **Managed Instances**.

3. Note the **Agent version**.

**To get the currently installed SSM Agent version from within the operating system**

**Windows**

1. Log in to your instance.
2. Run the following PowerShell command.

   ```powershell
   Get-WmiObject Win32_Product | Where-Object {$_._Name -eq 'Amazon SSM Agent'} | Select-Object Name,Version
   ```

**Linux**

   **Note**
   This command varies depending on the package manager for your operating system.

1. Log in to your instance.
2. Run the following command for Amazon Linux and Amazon Linux 2.

   ```bash
   yum info amazon-ssm-agent
   ```

We recommend using the latest version of the SSM Agent so you can benefit from new or updated capabilities. To ensure your managed instances are always running the most up-to-date version of the SSM Agent, you can automate the process of updating the SSM Agent. For more information, see **Automate updates to SSM Agent (p. 93)**.

**View SSM Agent logs**

SSM Agent writes information about executions, commands, scheduled actions, errors, and health statuses to log files on each instance. You can view log files by manually connecting to an instance, or you can automatically send logs to Amazon CloudWatch Logs. For more information about sending logs to CloudWatch, see **Monitoring AWS Systems Manager (p. 1134)**.

You can view SSM Agent logs on instances in the following locations.

**Linux**

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

**Windows**

```
%PROGRAMDATA%\Amazon\SSM\Logs\amazon-ssm-agent.log
%PROGRAMDATA%\Amazon\SSM\Logs\errors.log
```
For Linux instances, the SSM Agent stderr and stdout files are written to the following directory: /var/lib/amazon/ssm.

For information about enabling SSM Agent debug logging, see Enable SSM Agent debug logging (p. 91).

For more information about cihub/seelog configuration, see the Seelog Wiki on GitHub. For examples of cihub/seelog configurations, see the cihub/seelog examples repository on GitHub.

Enable SSM Agent debug logging

Use the following procedure to enable SSM Agent debug logging on Windows Server and Linux managed instances.

Linux

To enable SSM Agent debug logging on Linux instances

1. Either use Systems Manager Session Manager to connect to the instance where you want to enable debug logging, or log on to the managed instance. For more information, see Working with Session Manager (p. 812).
2. Make a copy of the seelog.xml.template file. Change the name of the copy to seelog.xml. The file is located in the following directory.
   `/etc/amazon/ssm/seelog.xml.template`
3. Edit the seelog.xml file to change the default logging behavior. Change the value of `minlevel` from `info` to `debug`, as shown in the following example.

   ```xml
   <seelog type="adaptive" mininterval="2000000" maxinterval="100000000" critmsgcount="500" minlevel="debug">
   </seelog>
   ```
4. Restart the SSM Agent using the following command.

   `sudo amazon-ssm-agent restart`

Windows

To enable SSM Agent debug logging on Windows Server instances

1. Either use Systems Manager Session Manager to connect to the instance where you want to enable debug logging, or log on to the managed instance. For more information, see Working with Session Manager (p. 812).
2. Make a copy of the seelog.xml.template file. Change the name of the copy to seelog.xml. The file is located in the following directory.
   `%PROGRAMFILES%\Amazon\SSM\seelog.xml.template`
3. Edit the seelog.xml file to change the default logging behavior. Change the value of `minlevel` from `info` to `debug`, as shown in the following example.

   ```xml
   <seelog type="adaptive" mininterval="2000000" maxinterval="100000000" critmsgcount="500" minlevel="debug">
   </seelog>
   ```
4. Locate the following entry:
Restrict access to root-level commands through SSM Agent

SSM Agent runs on EC2 instances using root permissions (Linux) or SYSTEM permissions (Windows Server). Because these are the highest level of system access privileges, any trusted entity that has been granted permission to send commands to SSM Agent has root or SYSTEM permissions. (In AWS, a trusted entity that can perform actions and access resources in AWS is called a principal. A principal can be an AWS account root user, an IAM user, or a role.)

This level of access is required for a principal to send authorized Systems Manager commands to SSM Agent, but also makes it possible for a principal to run malicious code by exploiting any potential vulnerabilities in SSM Agent.

In particular, permissions to run the commands SendCommand and StartSession should be carefully restricted. A good first step is to grant permissions for each command only to select principals in your organization. However, we recommend tightening your security posture even further by restricting which instances a principal can run these commands on. This can be done in the IAM user policy assigned to the principal. In the IAM policy, you can include a condition that limits the user to running commands only on instances that are tagged with specific Amazon EC2 tags, or combinations of EC2 tags.

For example, say you have two fleets of instances, one for testing, one for production. In the IAM policy applied to junior engineers, you specify that they can run commands only on instances tagged with ssm:resourceTag/testServer. But for a smaller group of lead engineers, who should have access to all instances, you grant access to instances tagged with both ssm:resourceTag/testServer and ssm:resourceTag/productionServer.

Using this approach, if junior engineers attempt to run a command on a production instance, they will be denied access because their assigned IAM policy does not provide explicit access to instances tagged with ssm:resourceTag/productionServer.

For more information and examples, see the following topics:

- Restricting Run Command access based on instance tags (p. 831)
- Restrict session access based on instance tags (p. 791)
Automate updates to SSM Agent

AWS releases a new version of SSM Agent when we add or update Systems Manager capabilities. If your instances use an older version of the agent, then you can't use the new capabilities or benefit from the updated capabilities. For these reasons, we recommend that you automate the process of updating SSM Agent on your instances using any of the following methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-click automated update on all instances (Recommended)</td>
<td>You can configure all instances in your AWS account to automatically check for and download new versions of SSM Agent. To do this, choose Agent auto update on the Managed instances page in the AWS Systems Manager console, as described later in this topic.</td>
</tr>
<tr>
<td>Global or selective update</td>
<td>You can use State Manager to create an association that automatically downloads and installs SSM Agent on your instances. If you want to limit the disruption to your workloads, you can create a Systems Manager maintenance window to perform the installation during designated time periods. Both methods enable you to create either a global update configuration for all of your instances or selectively choose which instances get updated. For information about creating a State Manager association, see Automatically update SSM Agent (CLI) (p. 908). For information about using a maintenance window, see Automatically Update SSM Agent (AWS CLI) and Automatically Update SSM Agent (Console).</td>
</tr>
<tr>
<td>Global or selective update for new environments</td>
<td>If you are getting started with Systems Manager, we recommend that you use the Update Systems Manager (SSM) Agent every two weeks option in Systems Manager Quick Setup. Quick Setup enables you to create either a global update configuration for all of your instances or selectively choose which instances get updated. For more information, see AWS Systems Manager Quick Setup (p. 16).</td>
</tr>
</tbody>
</table>

If you prefer to update SSM Agent on your instances manually, you can subscribe to notifications that AWS publishes when a new version of the agent is released. For information, see Subscribe to SSM Agent notifications (p. 94). After you subscribe to notifications, you can use Run Command to manually update one or more instances with the latest version. For more information, see Update SSM Agent by using Run Command (p. 835).

Automatically update SSM Agent

You can configure Systems Manager to automatically update SSM Agent on all managed instances in your AWS account. If you enable this option, then Systems Manager automatically checks every two weeks for a new version of the agent. If there is a new version, then Systems Manager automatically
updates the agent to the latest released version using the SSM document `AWS-UpdateSSMAgent`. We encourage you to choose this option to ensure that your instances are always running the most up-to-date version of SSM Agent.

**Note**
If you use a `yum` command to update SSM Agent on a managed instance after the agent has been installed or updated using the SSM document `AWS-UpdateSSMAgent`, you might see the following message: "Warning: RPMDB altered outside of yum." This message is expected and can be safely ignored.

**To automatically update SSM Agent**

2. In the navigation pane, choose Managed instances.
3. Choose Agent auto update.

To stop automatically deploying updated versions of SSM Agent to all managed instances in your account, choose Delete on the Managed instances page. This action deletes the State Manager association that automatically updates SSM Agent on your instances.

---

**Subscribe to SSM Agent notifications**

Amazon Simple Notification Service (Amazon SNS) can notify you when new versions of SSM Agent are released. Use the following procedure to subscribe to these notifications.

**Tip**
You can also subscribe to notifications by watching the SSM Agent Release Notes page on GitHub.

**To subscribe to SSM Agent notifications**

2. From the Region selector in the navigation bar, choose US East (N. Virginia), if it is not selected already. You must select this Region because the SNS notifications for SSM Agent that you are subscribing to are generated from this Region only.
3. In the navigation pane, choose Subscriptions.
4. Choose Create subscription.
5. For Create subscription, do the following:
   a. For Topic ARN, use the following Amazon Resource Name (ARN):
      ```
      arn:aws:sns:us-east-1:720620558202:SSM-Agent-Update
      ```
   b. For Protocol, choose Email or SMS.
   c. For Endpoint, type an email address that you can use to receive the notifications. If you choose SMS, type an area code and number.
   d. Choose Create subscription.
6. If you chose Email, you'll receive an email asking you to confirm your subscription. Open the email and follow the directions to complete your subscription.

Whenever a new version of SSM Agent is released, we send notifications to subscribers. If you no longer want to receive these notifications, use the following procedure to unsubscribe.
To unsubscribe from SSM Agent notifications

1. Open the Amazon SNS console.
2. In the navigation pane, choose Subscriptions.
3. Select the subscription and then choose Actions, Delete subscriptions. When prompted for confirmation, choose Delete.

About minimum S3 Bucket permissions for SSM Agent

This topic provides information about the Amazon Simple Storage Service (Amazon S3) buckets that SSM Agent might need to access in order to perform Systems Manager operations. These buckets are publicly accessible, but in some cases, you might need to provide explicit permission in an EC2 instance profile for Systems Manager, or in a service role for instances in a hybrid environment. Most commonly, you must grant these permissions if you are using a private VPC endpoint in your Systems Manager operations. Otherwise, your resources can’t access these public buckets.

To grant access to these buckets, you create a custom S3 permissions policy, and then attach it to your instance profile (for EC2 instances) or your service role (for on-premises servers and virtual machines (VMs) in a hybrid environment.

For SSM Agent updates, if the instance profile does not provide permissions to these buckets, the SSM Agent makes an HTTP call to download the update.

Note
These permissions only provide access to the AWS managed buckets required by SSM Agent. They don't provide the permissions that are necessary for other Amazon S3 operations. They also don't provide permission to your own S3 buckets.

For more information, see the following topics:

- Create an IAM instance profile for Systems Manager (p. 30)
- Create an IAM service role for a hybrid environment (p. 44)

Contents

- Required permissions (p. 95)
- Example (p. 96)

Required permissions

The following table describes each of the Amazon S3 policy permissions needed for using Systems Manager.

Amazon S3 permissions required by SSM Agent

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arn:aws:s3:::aws-ssm-region/*</td>
<td>Provides access to the S3 bucket containing modules required for use with SSM documents.</td>
</tr>
</tbody>
</table>

Note
In the Middle East (Bahrain) Region (me-south-1) only, this bucket uses a different
### Example

The following example illustrates how to provide access to the S3 buckets required for Systems Manager operations in the US East (Ohio) Region (us-east-2).

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

---

**region** represents the identifier for an AWS Region supported by AWS Systems Manager, such as us-east-2 for the US East (Ohio) Region. For a list of supported **region** values, see the Region column in Systems Manager service endpoints in the Amazon Web Services General Reference.
Example

```json
{
  "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-us-east-2/*",
    "arn:aws:s3:::amazon-ssm-packages-us-east-2/*",
    "arn:aws:s3:::us-east-2-birdwatcher-prod/*",
    "arn:aws:s3:::patch-baseline-snapshot-us-east-2/*"
  ]
}
```

**Important**

We recommend that you avoid using wildcard characters (*) in place of specific Regions in this policy. For example, use `arn:aws:s3:::aws-ssm-us-east-2/*` and do not use `arn:aws:s3:::aws-ssm-*/*`. Using wildcards could provide access to S3 buckets that you don't intend to grant access to. If you want to use the instance profile for more than one Region, we recommend repeating the first `Statement` block for each Region.
Partner and product integration

You can use AWS Systems Manager with partner and product technologies to automate the deployment, configuration, and maintenance of your managed instances.

Contents

- Referencing AWS Secrets Manager secrets from Parameter Store parameters (p. 98)
- Running scripts from GitHub and Amazon S3 (p. 101)
- Using Chef InSpec profiles with Systems Manager Compliance (p. 113)

Referencing AWS Secrets Manager secrets from Parameter Store parameters

Secrets Manager helps you organize and manage important configuration data such as credentials, passwords, and license keys. Parameter Store is now integrated with Secrets Manager so that you can retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters. These services include Amazon EC2, Amazon Elastic Container Service, AWS Lambda, AWS CloudFormation, AWS CodeBuild, AWS CodeDeploy, and other Systems Manager capabilities. By using Parameter Store to reference Secrets Manager secrets, you create a consistent and secure process for calling and using secrets in your code and configuration scripts.

For more information about Secrets Manager, see What Is AWS Secrets Manager? in the AWS Secrets Manager User Guide.

Important

Parameter Store functions as a pass-through service for references to Secrets Manager secrets. Parameter Store doesn't retain data or metadata about secrets. The reference is stateless.

Restrictions

Note the following restrictions when using Parameter Store to reference Secrets Manager secrets:

- You can only retrieve Secrets Manager secrets by using the GetParameter and GetParameters API actions. Modification operations and advance querying API actions, such as DescribeParameters or GetParametersByPath, are not supported for Secrets Manager.
- You can use the AWS CLI, AWS Tools for Windows PowerShell, and the SDKs to retrieve a secret by using Parameter Store.
- When you retrieve a Secrets Manager secret from Parameter Store, the parameter name must begin with the following reserved path: /aws/reference/secretsmanager/secretsmanager/secret_ID_in_Secrets_Manager.

Here is an example: /aws/reference/secretsmanager/CFCreds1
- Parameter Store honors IAM policies attached to Secrets Manager secrets. For example, if User 1 doesn't have access to Secret A, then User 1 can't retrieve Secret A by using Parameter Store.
- Parameters that reference Secrets Manager secrets can't use the Parameter Store versioning or history features.
- Parameter Store honors Secrets Manager version stages. If you reference a version stage, it can only use letters, numbers, a period (.), a hyphen (-), or an underscore (_). All other symbols specified in the version stage cause the reference to fail.
How to reference a Secrets Manager secret by using Parameter Store

The following procedure describes how to reference a Secrets Manager secret by using Parameter Store APIs. The procedure references other procedures in the AWS Secrets Manager User Guide.

**Note**
Before you begin, verify that you have permission to reference Secrets Manager secrets in Parameter Store parameters. If you have administrator privileges in Secrets Manager and Systems Manager, then you can reference or retrieve secrets by using Parameter Store APIs. If you reference a Secrets Manager secret in a Parameter Store parameter, and you don't have permission to access that secret, then the reference fails. For more information, see Authentication and Access Control for AWS Secrets Manager in the AWS Secrets Manager User Guide.

To reference a Secrets Manager secret by using Parameter Store

1. Create a secret in Secrets Manager. For more information, see Creating and Managing Secrets with AWS Secrets Manager.
2. Reference a secret by using the AWS CLI, AWS Tools for Windows PowerShell, or the SDK. When you reference a Secrets Manager secret, the parameter name must begin with the following reserved path: `/aws/reference/secretsmanager/`. By specifying this path, Systems Manager knows to retrieve the secret from Secrets Manager instead of Parameter Store. Here are some example parameters that correctly reference Secrets Manager secrets:

   • `/aws/reference/secretsmanager/CFCreds1`
   • `/aws/reference/secretsmanager/DBPass`

Here is a Java code example that references an access-key and a secret-key that are stored in Secrets Manager. This code example sets up an Amazon DynamoDB client. The code retrieves configuration data and credentials from Parameter Store. The configuration data is stored as a string parameter in Parameter Store and the credentials are stored in Secrets Manager. Even though the configuration data and credentials are stored in separate services, both sets of data can be access from Parameter Store by using the GetParameter API.

```java
/** *
* Initialize AWS System Manager Client with default credentials *
*/
AWSSimpleSystemsManagement ssm =
AWSSimpleSystemsManagementClientBuilder.defaultClient();
...

/** *
* Example method to launch DynamoDB client with credentials different from default *
* @return DynamoDB client *
*/
AmazonDynamoDB getDynamoDbClient() {
    //Getting AWS credentials from Secrets manager using GetParameter
    BasicAWSCredentials differentAWScreds = new BasicAWSCredentials(
        getParameter("/aws/reference/secretsmanager/access-key"),
        getParameter("/aws/reference/secretsmanager/secret-key"));

    //Initialize the DDB Client with different credentials
    final AmazonDynamoDB client = AmazonDynamoDBClient.builder()
        .withCredentials(new AWSStaticCredentialsProvider(differentAWScreds))
        .withRegion(getParameter("region")) //Getting config from Parameter Store
        .build();
```
Here are some AWS CLI examples.

**AWS CLI Example 1: Reference by using the name of the secret**

```bash
aws ssm get-parameter --name /aws/reference/secretsmanager/s1-secret --with-decryption
```

The command returns information like the following.

```
{
    "Parameter": {
        "Name": "/aws/reference/secretsmanager/s1-secret",
        "Value": "Fl*MEishm!al875",
        "Type": "SecureString",
        "LastModifiedDate": 2018-05-14T21:47:14.743Z,
        "ARN": "arn:aws:secretsmanager:us-west-1:123456789:secret:s1-secret-E18LRP",
        "SourceResult": {
            "CreatedDate": 1526343434.743,
            "Name": "s1-secret",
            "VersionId": "aaabbccc-1111-222-333-123456789",
            "SecretString": "Fl*MEishm!al875",
            "VersionStages": ["AWSCURRENT"],
            "ARN": "arn:aws:secretsmanager:us-west-1:123456789:secret:s1-secret-E18LRP"
        }
    }
}
```

**AWS CLI Example 2: Reference that includes the version ID**

```bash
aws ssm get-parameter --name /aws/reference/secretsmanager/s1-secret:11111-aaa-bbb-ccc-123456789 --with-decryption
```

The command returns information like the following.

```
{
    "Parameter": {
        "Name": "/aws/reference/secretsmanager/s1-secret",
        "Value": "Fl*MEishm!al875",
        "Type": "SecureString",
        "LastModifiedDate": 2018-05-14T21:47:14.743Z,
        "ARN": "arn:aws:secretsmanager:us-west-1:123456789:secret:s1-secret-E18LRP",
        "SourceResult":
```

```
Running scripts from GitHub and Amazon S3

This section describes how to use the AWS-RunRemoteScript pre-defined SSM document to download scripts from GitHub and Amazon S3, including Ansible Playbooks, Python, Ruby, and PowerShell scripts. By using this document, you no longer need to manually port scripts into Amazon EC2 or wrap them in SSM documents. Systems Manager integration with GitHub and Amazon S3 promotes infrastructure as code, which reduces the time it takes to manage instances while standardizing configurations across your fleet.

You can also create custom SSM documents that enable you to download and run scripts or other SSM documents from remote locations. For more information, see Creating composite documents (p. 1089).

Topics
- Running scripts from GitHub (p. 102)
- Running scripts from Amazon S3 (p. 107)
Running scripts from GitHub

This section describes how to download and run scripts from a private or public GitHub repository. You can run different types of scripts, including Ansible Playbooks, Python, Ruby, and PowerShell scripts.

You can also download a directory that includes multiple scripts. When you run the primary script in the directory, Systems Manager also runs any referenced scripts (as long as the referenced scripts are included in the directory).

Note the following important details about running scripts from GitHub.

• Systems Manager does not check to see if your script is capable of running on an instance. Before you download and run the script, you must verify that the required software is installed on the instance. Or, you can create a composite document that installs the software by using either Run Command or State Manager, and then downloads and runs the script.

• You are responsible for ensuring that all GitHub requirements are met. This includes refreshing your access token, as needed. You must also ensure that you don't surpass the number of authenticated or unauthenticated requests. For more information, see the GitHub documentation.

Topics
• Run Ansible Playbooks from GitHub (p. 102)
• Run Python scripts from GitHub (p. 104)

Run Ansible Playbooks from GitHub

This section includes procedures to help you run Ansible Playbooks from GitHub by using either the console or the AWS CLI.

Before You Begin

If you plan to run a script that is stored in a private GitHub repository, then you must create a Systems Manager SecureString parameter for your GitHub security access token. You can't access a script in a private GitHub repository by manually passing your token over SSH. The access token must be passed as a Systems Manager SecureString parameter. For more information about creating a SecureString parameter, see Creating Systems Manager parameters (p. 249).

Run an Ansible Playbook from GitHub (console)

Run an Ansible Playbook from GitHub


2. In the navigation pane, choose Run Command.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Run Command.

3. Choose Run command.


5. In Command parameters, do the following:

   • In Source Type, select GitHub.
In the **Source Info** box, type the required information to access the source in the following format:

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

This example downloads a file named `webserver.yml`.

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

**Note**

"branch" is required only if your SSM document is stored in a branch other than master.

To use the version of your scripts that are in a particular *commit* in your repository, use *commitID* with `getOptions` instead of *branch*. For example:

```json
"getOptions": "commitID:bbc1ddb94...b76d3bEXAMPLE",
```

6. In the **Command Line** field, type parameters for the script execution. Here is an example.

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

7. For **Other parameters**:

   - For **Comment**, type information about this command.
   - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. (Optional) For **Rate control**:

   - For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

**Note**

If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
• For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

9. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.

**Note**
The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task.
For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.
For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

11. Choose **Run**.

---

**Run an Ansible Playbook from GitHub by using the AWS CLI**

1. Install and configure the AWS CLI, if you have not already.
For information, see Install or upgrade AWS command line tools (p. 57).
2. Run the following command to download and run a script from GitHub.

```bash
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"]}
```

Here is an example.

```bash
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"]}
```

---

**Run Python scripts from GitHub**

This section includes procedures to help you run Python scripts from GitHub by using either the console or the AWS CLI.

**Run a Python script from GitHub (console)**

**Run a Python script from GitHub**

2. In the navigation pane, choose **Run Command**.

-or-

---

104
If the AWS Systems Manager home page opens first, choose the menu icon ( ) to open the navigation pane, and then choose Run Command.

3. Choose Run command.


5. For Command parameters, do the following:
   - In Source Type, select GitHub.
   - In the Source Info box, type the required information to access the source in the following format:

   ```json
   {
     "owner": "owner_name",
     "repository": "repository_name",
     "branch": "branch_name",
     "path": "path_to_document",
     "tokenInfo": "{{ssm-secure:SecureString_parameter_name}}"
   }
   ```

   For example, the following downloads a directory of scripts named complex-script:

   ```json
   {
     "owner": "TestUser1",
     "repository": "SSMTestDocsRepo",
     "branch": "myBranch",
     "path": "scripts/python/complex-script",
     "tokenInfo": "{{ssm-secure:myAccessTokenParam}}"
   }
   ```

   **Note**

   "branch" is required only if your scripts are stored in a branch other than master.

   To use the version of your scripts that are in a particular commit in your repository, use commitID with getOptions instead of branch. For example:

   ```json
   "getOptions": "commitID:bbc1ddb94...b76d3bEXAMPLE",
   ```

   - For Command Line, type parameters for the script execution. Here is an example.

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

     This example runs mainFile.py, which can then run other scripts in the complex-script directory.

   - (Optional) For Working Directory, type the name of a directory on the instance where you want to download and run the script.

   - (Optional) For Execution Timeout, specify the number of seconds for the system to wait before failing the script command execution.

6. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

   **Note**

   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.

7. For Other parameters:

   - For Comment, type information about this command.

   - For Timeout (seconds), specify the number of seconds for the system to wait before failing the overall command execution.
8. (Optional) For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

  **Note**
  
  If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

- For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

9. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.

  **Note**
  
  The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task.

  For more information, see **Create an IAM instance profile for Systems Manager** (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

    For more information about configuring Amazon SNS notifications for Run Command, see **Monitoring Systems Manager status changes using Amazon SNS notifications** (p. 1151).

11. Choose **Run**.

### Run a Python script from GitHub by using the AWS CLI

1. Install and configure the AWS CLI, if you have not already.

   For information, see **Install or upgrade AWS command line tools** (p. 57).

2. Run the following command to download and run a script from GitHub.

   ```bash
   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"]}'
   ```

   Here is an example.

   ```bash
   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"]}
   ```

   This example downloads a directory of scripts name complex-script. The **commandLine** entry runs mainFile.py, which can then run other scripts in the complex-script directory.
Running scripts from Amazon S3

This section describes how to download and run scripts from Amazon S3. You can run different types of scripts, including Ansible Playbooks, Python, Ruby, Shell, and PowerShell.

You can also download a directory that includes multiple scripts. When you run the primary script in the directory, Systems Manager also runs any referenced scripts (as long as the referenced scripts are included in the directory).

Note the following important details about running scripts from Amazon S3.

- Systems Manager does not check to see if your script is capable of running on an instance. Before you download and run the script, you must verify that the required software is installed on the instance. Or, you can create a composite document that installs the software by using either Run Command or State Manager, and then downloads and runs the script.
- Verify that your AWS Identity and Access Management (IAM) user account, role, or group has permission to read from the S3 bucket.
- Ensure that the instance profile on your EC2 instances has `s3:ListBucket` and `s3:GetObject` permissions. If the instance profile doesn't have these permissions, the system fails to download your script from the S3 bucket. For more information, see Using instance profiles in the IAM User Guide.

Topics
- Run Ruby scripts from Amazon S3 (p. 107)
- Run shell scripts from Amazon S3 (p. 109)
- Run a PowerShell script from Amazon S3 (p. 111)

Run Ruby scripts from Amazon S3

This section includes procedures to help you run Ruby scripts from Amazon S3 by using either the Systems Manager console or the AWS CLI.

Run a Ruby script from Amazon S3 (console)

Run a Ruby script from Amazon S3

2. In the navigation pane, choose Run Command.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (⋮) to open the navigation pane, and then choose Run Command.
3. Choose Run command.
5. In Command parameters, do the following:
   - In Source Type, select S3.
   - In the Source Info text box, type the required information to access the source in the following format:

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

   For example:
Run a Ruby script from Amazon S3 by using the AWS CLI

1. Install and configure the AWS CLI, if you have not already.
For information, see Install or upgrade AWS command line tools (p. 57).

2. Depending on the operating system type on your local machine, run one of the following commands to download and run a script from Amazon S3 (the Windows version includes the escape characters ("/"") you need to run the command from your command line tool):

**Windows** local machine:

```bash
aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=instance-IDs" --parameters "sourceType"="S3",sourceInfo='{"path":"https://s3.amazonaws.com/path_to_script\""}',"commandLine"="script_name_and_arguments"
```

Here is an example.

```bash
aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=i-1234567890abcdef0" --parameters "sourceType"="S3",sourceInfo='{"path":"https://s3.amazonaws.com/RubyTest/scripts/ruby/helloWorld.rb\""}',"commandLine"="helloWorld.rb argument-1 argument-2"
```

**Linux** local machine:

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

Here is an example.

```bash
aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=i-1234567890abcdef0" --parameters '{"sourceType": ["S3"],"sourceInfo": [{"path": "https://s3.amazonaws.com/RubyTest/scripts/ruby/helloWorld.rb\""} ],"commandLine": ["helloWorld.rb argument-1 argument-2"]}
```

---

**Run shell scripts from Amazon S3**

This section includes procedures to help you run Shell scripts from Amazon S3 by using either the Systems Manager console or the AWS CLI.

**Run a shell script from Amazon S3 (console)**

**Run a shell script from Amazon S3**

2. In the navigation pane, choose Run Command.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Run Command.
3. Choose Run command.
5. In Command parameters, do the following:
   - In Source Type, select S3.
• In the **Source Info** text box, type the required information to access the source in the following format:

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

For example:

```
{"path":"https://s3.amazonaws.com/shelltest/scripts/shell/helloWorld.sh"}
```

• In the **Command Line** field, type parameters for the script execution. Here is an example.

```
helloWorld.sh argument-1 argument-2
```

• (Optional) In the **Working Directory** field, type the name of a directory on the instance where you want to download and run the script.

• (Optional) In **Execution Timeout**, specify the number of seconds for the system to wait before failing the script command execution.

6. In the **Targets** section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

**Note**

If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.

7. For **Other parameters**:

• For **Comment**, type information about this command.

• For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. (Optional) For **Rate control**:

• For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

  **Note**
  
  If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

• For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

9. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.

  **Note**
  
  The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

Run a shell script from Amazon S3 by using the AWS CLI

1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Depending on the operating system type on your local machine, run one of the following commands to download and run a script from Amazon S3 (the Windows version includes the escape characters ("\") you need to run the command from your command line tool):

   **Windows** local machine:

   ```bash
   aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=instance-IDs" --parameters "sourceType="S3",sourceInfo='{"path":":https://s3.amazonaws.com/path_to_script\"}'","commandLine="script_name_and_arguments"
   ```

   Here is an example.

   ```bash
   aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=i-1234567890abcdef0" --parameters "sourceType="S3",sourceInfo='{"path":":https://s3.amazonaws.com/ShellTest/scripts/shell/helloWorld.sh\"}'","commandLine="helloWorld.sh argument-1 argument-2"
   ```

   **Linux** local machine:

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

   Here is an example.

   ```bash
   aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=i-1234567890abcdef0" --parameters '{"sourceType":"S3","sourceInfo": [{"path":":https://s3.amazonaws.com/ShellTest/scripts/shell/helloWorld.sh\"}]},{"commandLine": [{"helloWorld.sh argument-1 argument-2"]}
   ```

Run a PowerShell script from Amazon S3

This section includes procedures to help you run PowerShell scripts from Amazon S3 by using either the EC2 console or the AWS CLI.

Run a PowerShell script from Amazon S3 (console)

**Run a PowerShell script from Amazon S3**

2. In the navigation pane, choose Run Command.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Run Command.
3. Choose Run command.
5. In Command parameters, do the following:
   - In Source Type, select S3.
   - In the Source Info text box, type the required information to access the source in the following format:
     ```json
     {"path": "https://s3.amazonaws.com/path_to_script"}
     ```
     For example:
     ```json
     {"path": "https://s3.amazonaws.com/PowerShellTest/powershell/helloPowershell.ps1"}
     ```
   - In the Command Line field, type parameters for the script execution. Here is an example.
     ```bash
     helloPowershell.ps1 argument-1 argument-2
     ```
   - (Optional) In the Working Directory field, type the name of a directory on the instance where you want to download and run the script.
   - (Optional) In Execution Timeout, specify the number of seconds for the system to wait before failing the script command execution.
6. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.
   **Note**
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.
7. For Other parameters:
   - For Comment, type information about this command.
   - For Timeout (seconds), specify the number of seconds for the system to wait before failing the overall command execution.
8. (Optional) For Rate control:
   - For Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.
     **Note**
     If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
   - For Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.
9. (Optional) For Output options, to save the command output to a file, select the Write command output to an S3 bucket box. Type the bucket and prefix (folder) names in the boxes.
   **Note**
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.
10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

11. Choose **Run**.

### Run a PowerShell script from S3 by using the AWS CLI

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Depending on the operating system type on your local machine, run one of the following commands to download and run a script from Amazon S3 (the Windows version includes the escape characters (\") you need to run the command from your command line tool):

   **Windows** local machine:

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

   Here is an example.

   ```
   aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=i-1234567890abcdef0" --parameters "sourceType="S3",sourceInfo='{"path":"https://s3.amazonaws.com/PowerShellTest/scripts/powershell/helloWorld.ps1\""'},"commandLine"="helloWorld.ps1 argument-1 argument-2"
   ```

   **Linux** local machine:

   ```
   aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=instance-IDs" --parameters '{"sourceType":"S3"},"sourceInfo": ["{"path":"https://s3.amazonaws.com/PowerShellTest/scripts/powershell/helloWorld.ps1\""}],"commandLine": ["script_name_and_arguments"]
   ```

   Here is an example.

   ```
   aws ssm send-command --document-name "AWS-RunRemoteScript" --targets "Key=instanceids,Values=i-1234567890abcdef0" --parameters '{"sourceType":"S3"},"sourceInfo": ["{"path":"https://s3.amazonaws.com/PowerShellTest/scripts/powershell/helloWorld.ps1\""}],"commandLine": ["helloWorld.ps1 argument-1 argument-2"]
   ```

### Using Chef InSpec profiles with Systems Manager Compliance

Systems Manager now integrates with **Chef InSpec**. InSpec is an open-source testing framework that enables you to create human-readable profiles to store in GitHub or Amazon S3. Then you can use Systems Manager to run compliance scans and view compliant and noncompliant instances. A **profile** is a security, compliance, or policy requirement for your computing environment. For example, you can
create profiles that perform the following checks when you scan your instances with Systems Manager Compliance:

- Check if specific ports are open or closed.
- Check if specific applications are running.
- Check if certain packages are installed.
- Check Windows Registry keys for specific properties.

You can create InSpec profiles for EC2 instances and on-premises servers or virtual machines (VMs) that you manage with Systems Manager. The following sample Chef InSpec profile checks to see if port 22 is open.

```ruby
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 includes a collection of resources that help you quickly write checks and auditing controls. InSpec uses the InSpec Domain-specific Language (DSL) for writing these controls in Ruby. You can also use profiles created by a large community of InSpec users. For example, the DevSec chef-os-hardening project on GitHub includes dozens of profiles to help you secure your instances and servers. You can author and store profiles in GitHub or Amazon Simple Storage Service (Amazon S3).

**How it works**

Here is how the process of using InSpec profiles with Systems Manager Compliance works.

1. Either identify predefined InSpec profiles that you want to use, or create your own. You can use predefined profiles on GitHub to get started. For information about how to create your own InSpec profiles, see Chef InSpec Profiles.
2. Store profiles in either a public or private GitHub repository, or in an S3 bucket.
3. Run Compliance with your InSpec profiles by using the AWS-RunInspecChecks SSM document. You can begin a Compliance scan by using Run Command (for on-demand scans), or you can schedule regular Compliance scans by using State Manager.
4. Identify noncompliant instances by using the Compliance API or the Systems Manager Compliance console.

**Note**

Chef uses a client on your instances to process the profile. You don't need to install the client. When Systems Manager runs the AWS-RunInspecChecks SSM document, the system checks if the client is installed. If not, Systems Manager installs the Chef client during the scan, and then uninstalls the client after the scan is completed.

**Running an InSpec compliance scan**

This section includes information about how to run an InSpec Compliance scan by using the Systems Manager console and the AWS CLI. The console procedure shows you how to configure State Manager to run the scan. The AWS CLI procedure shows you how to configure Run Command to run the scan.
Running an InSpec compliance scan with State Manager by using the console

To run an InSpec compliance scan with State Manager by using the AWS Systems Manager console

2. In the navigation pane, choose State Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose State Manager.
3. Choose Create association.
4. In the Provide association details section, enter a name.
5. In the Document list, choose AWS-RunInspecChecks.
6. In the Document version list, choose Latest at runtime.
7. In the Parameters section, in the Source Type list, choose either GitHub or S3.

   If you choose GitHub, then enter the path to an InSpec profile in either a public or private GitHub repository in the Source Info field. Here is an example path to a public profile provided by the Systems Manager team from the following location: https://github.com/awslabs/amazon-ssm/tree/master/Compliance/InSpec/PortCheck.

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

   If you choose S3, then enter a valid URL to an InSpec profile in an S3 bucket in the Source Info field.

   For more information about how Systems Manager integrates with GitHub and Amazon S3, see Running scripts from GitHub and Amazon S3 (p. 101).

8. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

   Note
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.

9. In the Specify schedule section, use the schedule builder options to create a schedule for when you want the Compliance scan to run.
10. (Optional) For Rate control:

    - For Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.

      Note
      If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

    - For Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

11. (Optional) For Output options, to save the command output to a file, select the Write command output to an S3 bucket box. Type the bucket and prefix (folder) names in the boxes.
Running an InSpec compliance scan

12. Choose **Create Association**. The system creates the association and automatically runs the Compliance scan.

13. Wait several minutes for the scan to complete, and then choose **Compliance** in the navigation pane.

14. In **Corresponding managed instances**, locate instances where the **Compliance Type** column is **Custom:Inspec**.

15. Choose an instance ID to view the details of noncompliant statuses.

Running an InSpec compliance scan with Run Command by using the AWS CLI

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run one of the following commands to run an InSpec profile from either GitHub or Amazon S3.

   The command takes the following parameters:

   - **sourceType**: GitHub or Amazon S3
   - **sourceInfo**: URL to the InSpec profile folder either in GitHub or an S3 bucket. The folder must contain the base InSpec file (*.yml) and all related controls (*.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"}"]}
   ```

   Here is an example.

   ```
   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"}"]}
   ```

   Here is an example.

   ```
   aws ssm send-command --document-name "AWS-RunInspecChecks" --targets
   '[("Key": "tag:testEnvironment", "Values": ["webServers"])]' --parameters
   '{"sourceType": ["S3"], "sourceInfo": ["{"path": "https://s3.amazonaws.com/directory/Inspec.yml_file"}"]}
   ```
3. Run the following command to view a summary of the Compliance scan.

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

4. Run the following command to drill down into an instance that is not compliant.

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

**Related AWS Services**

The following related services can help you assess Compliance and work with Chef.

- **Amazon Inspector** lets you perform security assessments on your instances based on and common vulnerabilities described in Central Internet Security (CIS) standards.
- **AWS OpsWorks for Chef Automate** lets you run a Chef Automate server in AWS.
Operations Management

Operations Management is a suite of capabilities that help you manage your AWS resources.

Topics
- AWS Systems Manager Explorer (p. 118)
- AWS Systems Manager OpsCenter (p. 138)
- Amazon CloudWatch dashboards hosted by Systems Manager (p. 168)
- Trusted Advisor and Personal Health Dashboards hosted by Systems Manager (p. 168)

AWS Systems Manager Explorer

AWS Systems Manager Explorer is a customizable operations dashboard that reports information about your AWS resources. Explorer displays an aggregated view of operations data (OpsData) for your AWS accounts and across Regions. In Explorer, OpsData includes metadata about your EC2 instances, patch compliance details, and operational work items (OpsItems). Explorer provides context about how OpsItems are distributed across your business units or applications, how they trend over time, and how they vary by category. You can group and filter information in Explorer to focus on items that are relevant to you and that require action. When you identify high priority issues, you can use Systems Manager OpsCenter to run Automation runbooks and quickly resolve those issues.

The following image shows some of the individual report boxes, called widgets, which are available in Explorer.
What are the features of Explorer?

Explorer includes the following features:

- **Customizable display of actionable information**: Explorer includes drag-and-drop widgets that automatically display actionable information about your AWS resources. Explorer displays information in two types of widgets.

- **Informational widgets**: These widgets summarize data from Amazon EC2 and Systems Manager Patch Manager. These widgets provide important context to help you understand the state and operational risks of your AWS resources. Examples of informational widgets include **Instance count**, **Managed instances**, **Instance by AMI**, and **Non-compliant instances for patching**.

- **OpsItem widgets**: A Systems Manager OpsItem is an operational work item that is related to one or more AWS resources. OpsItems are a feature of Systems Manager OpsCenter. OpsItems may require DevOps engineers to investigate and potentially remediate an issue. Examples of possible OpsItems include high EC2 instance CPU utilization, detached Amazon Elastic Block Store volumes, AWS CodeDeploy deployment failure, or Systems Manager Automation execution failure. Examples of OpsItem widgets include **Open OpsItem summary**, **OpsItem by status**, and **OpsItems over time**.

- **Filters**: Each widget offers the ability to filter information based on AWS account, Region, and tag. Filters help you quickly refine the information displayed in Explorer.

- **Direct links to service screens**: To help you investigate issues with AWS resources, Explorer widgets contain direct links to related service screens. Filters applied to a widget remain in effect if you navigate to a related service screen.

- **Groups**: To help you understand the types of operational issues across your organization, some widgets enable you to group data based on account, Region, and tag.

- **Reporting tag keys**: When you set up Explorer, you can specify up to five tag keys. These keys help you group and filter data in Explorer. If a specified key matches a key on a resource that generates an OpsItem, then the key and value are included in the OpsItems.

- **Three modes of AWS account and Region display**: Explorer includes the following display modes for OpsData and OpsItems in AWS accounts and Regions:
  1. **Single-account/single-Region**: This is the default view. This mode enables users to view data and OpsItems from their own account and the current Region.
  2. **Single-account/multiple-Region**: This mode requires you to create one or more resource data syncs by using the Explorer Settings page. A resource data sync aggregates OpsData from one or more Regions. After you create a resource data sync, you can toggle which sync to use on the Explorer dashboard. You can then filter and group data based on Region.
  3. **Multiple-account/multiple-Region**: This mode requires that your organization or company use AWS Organizations with All features enabled. After you configure AWS Organizations in your computing environment, you can aggregate all account data in a master account. You can then create resource data syncs so that you can filter and group data based on Region. For more information about Organizations All features mode, see Enabling All Features in Your Organization.

- **Reporting**: You can export Explorer reports as comma separated value (.csv) files to an Amazon Simple Storage Service (Amazon S3) bucket. You receive an alert from Amazon Simple Notification Service (Amazon SNS) when an export completes.

How does Explorer relate to OpsCenter?

Systems Manager OpsCenter (p. 138) provides a central location where operations engineers and IT professionals view, investigate, and resolve OpsItems related to AWS resources. Explorer is a report hub where DevOps managers view aggregated summaries of their operations data, including OpsItems, across AWS Regions and accounts. Explorer helps users discover trends and patterns and, if necessary, quickly resolve issues using Systems Manager Automation runbooks.
OpsCenter setup is now integrated with Explorer Setup. If you already set up OpsCenter, then Explorer automatically displays operations data, including aggregated information about OpsItems. If you have not set up OpsCenter, then you can use Explorer Setup to get started with both capabilities. For more information, see Getting started with Systems Manager Explorer and OpsCenter (p. 121).

What is OpsData?

OpsData is any operations data that is displayed in the Systems Manager Explorer dashboard. Explorer retrieves OpsData from the following sources:

- **Amazon Elastic Compute Cloud (Amazon EC2)**
  
  Data displayed in Explorer includes: total number of instances, total number of managed and unmanaged instances, and a count of instances using a specific Amazon Machine Image (AMI).

- **Systems Manager OpsCenter**
  
  Data displayed in Explorer includes: a count of OpsItems by status, a count of OpsItems by severity, a count of open OpsItems across groups and across 30-day time periods, and historical data of OpsItems over time.

- **Systems Manager Patch Manager**
  
  Data displayed in Explorer includes a count of instances that aren't patch compliant.

- **AWS Trusted Advisor**
  
  Data displayed in Explorer includes: status of best practice checks for EC2 reserved instances in the areas of cost optimization, security, fault tolerance, performance, and service limits.

- **AWS Compute Optimizer**
  
  Data displayed in Explorer includes: a count of Under provisioned and Over provisioned EC2 instances, optimization findings, on-demand pricing details, and recommendations for instance type and price.

You can view and manage OpsData sources from the Explorer Settings page. For information about setting up and configuring services that populate Explorer widgets with OpsData, see Setting up related services (p. 121).

Is there a charge to use Explorer?

Yes. When you enable the default rules for creating OpsItems during Integrated Setup, you initiate a process that automatically creates OpsItems. Your account is charged based on the number of OpsItems created per month. Your account is also charged based on the number of GetOpsItem, DescribeOpsItem, UpdateOpsItem, and GetOpsSummary API calls made per month. Additionally, you can be charged for public API calls to other services that expose relevant diagnostic information. For more information, see AWS Systems Manager Pricing.

Is Explorer available in all AWS regions?

No. Explorer is not available in the BAH (me-south-1), HKG (ap-east-1), and CPT (af-south-1) Regions.

Topics

- Getting started with Systems Manager Explorer and OpsCenter (p. 121)
- Using Systems Manager Explorer (p. 131)
- Exporting OpsData from Systems Manager Explorer (p. 134)
Getting started with Systems Manager Explorer and OpsCenter

Systems Manager uses an integrated setup experience to help you get started with Systems Manager Explorer and Systems Manager OpsCenter. In this documentation, Explorer and OpsCenter Setup is called Integrated Setup. If you already set up OpsCenter, you still need to complete Integrated Setup to verify settings and options. If you have not set up OpsCenter, then you can use Integrated Setup to get started with both capabilities. Integrated Setup performs the following tasks.

1. **Configures roles and permissions (p. 122):** Integrated Setup creates an AWS Identity and Access Management (IAM) role that enables Amazon CloudWatch Events to automatically create OpsItems based on default rules. After setting up, you must configure IAM user, group, or role permissions for OpsCenter, as described in this section.

2. **Enables default rules for OpsItem creation (p. 126):** Integrated Setup creates default rules in CloudWatch Events. These rules automatically create OpsItems in response to events. Examples of these events are: state change for an AWS resource, a change in security settings, or a service becoming unavailable.

3. **Enables OpsData sources (p. 126):** Integrated Setup enables data sources that populate Explorer widgets.

4. ** Enables you to specify reporting tag keys (p. 127):** Integrated Setup enables you to specify up to five reporting tag keys to automatically assign to new OpsItems that meet specific criteria.

After you complete Integrated Setup, we recommend that you **Set up Explorer to display data from multiple Regions and accounts (p. 127).** Explorer and OpsCenter automatically synchronize OpsData and OpsItems for the AWS account and Region you used when you completed Integrated Setup. You can aggregate OpsData and OpsItems from other accounts and Regions by creating a resource data sync.

**Note**
You can change setup configurations at any time on the **Settings** page.

### Setting up related services

Explorer and OpsCenter collect information from, or interact with, other AWS services and Systems Manager capabilities. We recommend that you set up and configure these other services or capabilities before you use Integrated Setup.

The following table includes tasks that enable Explorer and OpsCenter to collect information from, or interact with, other AWS services and Systems Manager capabilities.

<table>
<thead>
<tr>
<th>Task</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify permissions in Systems Manager Automation</td>
<td>Explorer and OpsCenter enable you to remediate issues with AWS resources by using Systems Manager Automation documents (runbooks). To use this remediation capability, you must have permission to run Systems Manager Automation documents. For more information, see <strong>Getting started with Automation (p. 281).</strong></td>
</tr>
<tr>
<td>Set up and configure Systems Manager Patch Manager</td>
<td>Explorer includes a widget that provides information about patch compliance. To view this</td>
</tr>
</tbody>
</table>
Getting started

<table>
<thead>
<tr>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
</tr>
<tr>
<td>Data in Explorer, you must configure patching. For more information, see AWS Systems Manager Patch Manager (p. 911).</td>
</tr>
<tr>
<td>Enable AWS Config Configuration Recorder</td>
</tr>
<tr>
<td>Explorer uses data provided by AWS Config configuration recorder to populate widgets with information about your EC2 instances. To view this data in Explorer, enable AWS Config configuration recorder. For more information, see Managing the Configuration Recorder.</td>
</tr>
<tr>
<td>Note</td>
</tr>
<tr>
<td>After you enable configuration recorder, Systems Manager can take up to six hours to display data in Explorer widgets that display information about your EC2 instances.</td>
</tr>
<tr>
<td>Enable AWS Trusted Advisor</td>
</tr>
<tr>
<td>Explorer uses data provided by Trusted Advisor to display a status of best practice checks for Amazon EC2 reserved instances in the areas of cost optimization, security, fault tolerance, performance, and service limits. To view this data in Explorer, you must have a business or enterprise support plan. For more information, see AWS Support.</td>
</tr>
<tr>
<td>Enable AWS Compute Optimizer</td>
</tr>
<tr>
<td>Explorer uses data provided by Compute Optimizer to display details a count of Under provisioned and Over provisioned EC2 instances, optimization findings, on-demand pricing details, and recommendations for instance type and price. To view this data in Explorer, enable Compute Optimizer. For more information, see Getting started with AWS Compute Optimizer.</td>
</tr>
</tbody>
</table>

Configuring roles and permissions for Systems Manager Explorer

Integrated Setup automatically creates and configures IAM roles for Systems Manager Explorer and OpsCenter. If you completed Integrated Setup, then you don't need to perform any additional tasks to configure roles and permissions for Explorer. However, you must configure permission for OpsCenter, as described later in this topic.

Contents

- About the roles created by integrated setup (p. 122)
- Configuring permissions for Systems Manager OpsCenter (p. 123)

About the roles created by integrated setup

Integrated Setup creates and configures the following roles for working with Explorer and OpsCenter.

- **AWSServiceRoleForAmazonSSM**: Provides access to AWS Resources managed or used by Systems Manager.
• **OpsItem-CWE-Role**: Enables CloudWatch Events to create OpsItems in response to common events.

• **AWSServiceRoleForAmazonSSM_AccountDiscovery**: Enables Systems Manager to call other AWS services to discover AWS account information when synchronizing data. For more information about this role, see About the AWSServiceRoleForAmazonSSM_AccountDiscovery role (p. 123).

• **AmazonSSMExplorerExport**: Enables Explorer to export OpsData to a comma-separated value (CSV) file.

**About the AWSServiceRoleForAmazonSSM_AccountDiscovery role**

If you configure Explorer to display data from multiple accounts and Regions by using AWS Organizations and a resource data sync, then Systems Manager creates a service-linked role. Systems Manager uses this role to get information about your AWS accounts in AWS Organizations. The role uses the following permissions policy.

```json
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect":"Allow",
         "Action": [
            "organizations:DescribeAccount",
            "organizations:DescribeOrganization",
            "organizations:ListAccounts",
            "organizations:ListAWSServiceAccessForOrganization",
            "organizations:ListChildren",
            "organizations:ListParents"
         ],
         "Resource": "*"
      }
   ]
}
```

For more information about the AWSServiceRoleForAmazonSSM_AccountDiscovery role, see Using Roles to Collect AWS Account Information for Systems Manager Explorer (p. 1125).

**Configuring permissions for Systems Manager OpsCenter**

After you complete Integrated Setup, you must configure IAM user, group, or role permissions so that users can perform actions in OpsCenter.

**Before You Begin**

OpsItems can only be viewed or edited in the account where they were created. You can't share or transfer OpsItems across AWS accounts. For this reason, we recommend that you configure permissions for OpsCenter in the AWS account that is used to run your AWS workloads. You can then create IAM users or groups in that account. In this way, multiple operations engineers or IT professionals can create, view, and edit OpsItems in the same AWS account.

Explorer and OpsCenter use the following API actions. You can use all features of Explorer and OpsCenter if your IAM user, group, or role has access to these actions. You can also create more restrictive access, as described later in this section.

• `CreateOpsItem`
• `CreateResourceDataSync`
• `DescribeOpsItems`
• `DeleteResourceDataSync`
The following procedure describes how to add a full-access inline policy to an IAM user. If you prefer, you can specify read-only permission by assigning the following inline policy to a user’s account, group, or role.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem",
        "ssm:GetOpsSummary",
        "ssm:DescribeOpsItems"
      ],
      "Resource": "*"
    }
  ]
}
```

For more information about creating and editing IAM policies, see Creating IAM Policies in the IAM User Guide. For information about how to assign this policy to an IAM group, see Attaching a Policy to an IAM Group.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users.
3. In the list, choose a name.
4. Choose the Permissions tab.
5. On the right side of the page, under Permission policies, choose Add inline policy.
6. Choose the JSON tab.
7. Replace the default content with the following:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsItem",
        "ssm:GetOpsSummary",
        "ssm:DescribeOpsItems",
        "ssm:CreateOpsItem",
        "ssm:CreateResourceDataSync",
        "ssm:DeleteResourceDataSync",
        "ssm:ListResourceDataSync",
        "ssm:UpdateResourceDataSync"
      ],
      "Resource": "*"
    }
  ]
}
```
9. On the Review policy page, for Name, enter a name for the inline policy. For example: OpsCenter-Access-Full.
10. Choose Create policy.

Restricting access to OpsItems by using tags

You can also restrict access to OpsItems by using an inline IAM policy that specifies tags. The policy uses the following format.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "One_or_more_OpsItem_API_actions"
      ],
      "Resource": "*",
      "Condition": { "StringEquals": { "ssm:resourceTag/tag_key": "tag_value" } }
    }
  ]
}
```

Here is an example that specifies a tag key of Department and a tag value of Finance. With this policy, the user can only call the GetOpsItem API action to view OpsItems that were previously tagged with Key=Department and Value=Finance. Users can't view any other OpsItems.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ssm:GetOpsItem"],
      "Resource": "*",
      "Condition": { "StringEquals": { "ssm:resourceTag/Department": "Finance" } }
    }
  ]
}
```

Here is an example that specifies API actions for viewing and updating OpsItems. This policy also specifies two sets of tag key-value pairs: Department-Finance and Project-Unity.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ssm:GetOpsItem", "ssm:UpdateOpsItem"],
      "Resource": "*",
      "Condition": { "StringEquals": { "ssm:resourceTag/Department": "Finance" } }
    }
  ]
}
```
"Resource": "*",
"Condition": {
    "StringEquals": {
        "ssm:resourceTag/Department": "Finance",
        "ssm:resourceTag/Project": "Unity"
    }
}
}

For information about adding tags to an OpsItem, see Creating OpsItems manually (p. 150).

**Enabling default rules**

Integrated Setup automatically configures the following default rules in CloudWatch Events. These rules create OpsItems in Systems Manager OpsCenter. If you don't want CloudWatch Events to create OpsItems for the following events, then clear this option in Integrated Setup. If you prefer, you can specify OpsCenter as the target of specific CloudWatch Events events. For more information, see Configuring CloudWatch Events to automatically create OpsItems for specific events (p. 149). You can also disable the default rules at any time on the Settings page.

**Important**
Currently, you can't edit the **Category** and **Severity** values for default rules but you can edit these values on OpsItems created from the default rules.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Category</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSMOperstems-Autoscaling-instance-launch-failure</td>
<td>Availability</td>
<td>2-High</td>
</tr>
<tr>
<td>SSMOperstems-Autoscaling-instance-termination-failure</td>
<td>Availability</td>
<td>2-High</td>
</tr>
<tr>
<td>SSMOperstems-EBS-snapshot-copy-failed</td>
<td>Availability</td>
<td>2-High</td>
</tr>
<tr>
<td>SSMOperstems-EBS-snapshot-creation-failed</td>
<td>Availability</td>
<td>2-High</td>
</tr>
<tr>
<td>SSMOperstems-EBS-volume-performance-issue</td>
<td>Performance</td>
<td>3-Medium</td>
</tr>
<tr>
<td>SSMOperstems-EC2-issue</td>
<td>Availability</td>
<td>2-High</td>
</tr>
<tr>
<td>SSMOperstems-EC2-scheduled-change</td>
<td>Availability</td>
<td>3-Medium</td>
</tr>
<tr>
<td>SSMOperstems-RDS-issue</td>
<td>Availability</td>
<td>2-High</td>
</tr>
<tr>
<td>SSMOperstems-RDS-scheduled-change</td>
<td>Availability</td>
<td>3-Medium</td>
</tr>
<tr>
<td>SSMOperstems-SSM-maintenance-window-execution-failed</td>
<td>Availability</td>
<td>3-Medium</td>
</tr>
<tr>
<td>SSMOperstems-SSM-maintenance-window-execution-timesout</td>
<td>Availability</td>
<td>2-High</td>
</tr>
</tbody>
</table>
Specifying tag keys

When you set up Systems Manager Explorer, you can specify up to five reporting tag keys. These tag keys should already exist on your AWS resources. These are not new tag keys. After adding the keys to the system, you can then filter OpsItems in Explorer by using these tag keys.

Note
You can also specify reporting tag keys on the Settings page.

Setting up Systems Manager Explorer to display data from multiple accounts and Regions

Systems Manager uses an integrated setup experience to help you get started with Systems Manager Explorer and Systems Manager OpsCenter. After completing Integrated Setup, Explorer and OpsCenter automatically synchronize data. More specifically, these capabilities synchronize OpsData and OpsItems for the AWS account and Region you used when you completed Integrated Setup. If you want to aggregate OpsData and OpsItems from other accounts and Regions, you must create a resource data sync, as described in this topic.

Note
For more information about Integrated Setup, see Getting started with Systems Manager Explorer and OpsCenter (p. 121).

About Resource Data Sync for Explorer

Resource data sync for Explorer offers two aggregation options:

- **Single-account/Multiple-regions**: You can configure Explorer to aggregate OpsItems and OpsData data from multiple AWS Regions, but the data set is limited to the current AWS account.

- **Multiple-accounts/Multiple-regions**: You can configure Explorer to aggregate data from multiple AWS Regions and accounts. This option requires that you set up and configure AWS Organizations. After you set up and configure AWS Organizations, you can aggregate data in Explorer by organizational unit (OU) or for an entire organization. Systems Manager aggregates the data into the AWS Organizations master account before displaying it in Explorer. For more information, see What is AWS Organizations? in the AWS Organizations User Guide.

The following diagram shows a resource data sync configured to work with AWS Organizations. In this scenario, the user has two accounts defined in AWS Organizations. Resource data sync aggregates data from both accounts and multiple AWS Regions into the AWS Organizations master account where it is then displayed in Explorer.
Before You Begin

Note the following important details about resource data sync for Explorer.

- Explorer supports a maximum of five resource data syncs.
- You must complete Integrated Setup in each account and Region where you want Explorer to access data. If you don’t, Explorer won’t display OpsData and OpsItems for those accounts and Regions in which you didn’t complete Integrated Setup.
- To view OpsData and OpsItems from multiple accounts, you must have the AWS Organizations All features mode enabled and you must be signed into the AWS Organizations Master Account.
- After you create a resource data sync for a Region, you can’t change the account options for that sync. For example, if you create a sync in the us-east-2 (Ohio) Region and you choose the Include only the current account option, you can’t edit that sync later and choose the Include all accounts from my AWS Organizations configuration option. Instead, you must delete the first resource data sync, and create a new one. For more information, see Deleting a Systems Manager Explorer Resource Data Sync (p. 134)
- OpsData viewed in Explorer is read-only.

Use the following procedure to create a resource data sync for Explorer.

**To create a resource data sync**

2. In the navigation pane, choose Explorer.
3. Choose Settings.
4. In the **Configure resource data sync** section, choose **Create resource data sync**.
5. For **Resource data sync name**, enter a name.
6. In the **Add accounts** section, choose an option.

   **Note**
   To use either of the AWS Organizations options, you must be logged into the AWS Organizations master account or you must be logged into an Explorer delegated administrator account. For more information about the delegated administrator account, see [Configuring a Delegated Administrator (p. 129)](https://aws.amazon.com/documentation/systemsmanger/getting-started#configure-resource-data-sync).

7. In the **Regions to include** section, choose one of the following options.
   - Choose **All current and future regions** to automatically sync data from all current AWS Regions and any new Regions that come online in the future.
   - Choose **All regions** to automatically sync data from all current AWS Regions.
   - Individually choose Regions that you want to include.
8. Choose **Create resource data sync**.

The system can take several minutes to populate Explorer with data after you create a resource data sync. You can view the sync by choosing it from the **Select a resource data sync** list in Explorer.

### Configuring a Delegated Administrator

If you aggregate Explorer data from multiple AWS Regions and accounts by using resource data sync with AWS Organizations, then we suggest that you configure a delegated administrator for Explorer. A delegated administrator improves Explorer security in the following ways.

- You limit the number of Explorer administrators who can create or delete multi-account and Region resource data syncs to only one individual.
- You no longer need to be logged into the AWS Organizations master account to administer resource data syncs in Explorer.


### Topics
- **Before you begin (p. 129)**
- **Configure an Explorer delegated administrator (p. 130)**
- **Deregister an Explorer delegated administrator (p. 131)**

### Before you begin

The following list includes important information about Explorer delegated administration.

- You can delegate only one account for Explorer administration.
- The account ID that you specify as an Explorer delegated administrator must be listed as a member account in AWS Organizations. For more information, see [Creating an AWS account in your organization](https://aws.amazon.com/documentation/organizations/user-guide) in the [AWS Organizations User Guide](https://aws.amazon.com/documentation/organizations/user-guide).
- A delegated administrator can use all Explorer resource data sync API actions in the console or by using programmatic tools such as the SDK, the AWS CLI, or AWS Tools for Windows PowerShell. Resource data sync API actions include the following: `CreateResourceDataSync`, `DeleteResourceDataSync`, `ListResourceDataSync`, and `UpdateResourceDataSync`.  

---

129
A delegated administrator can search, filter, and aggregate Explorer data in the console or by using programmatic tools such as the SDK, the AWS CLI, or AWS Tools for Windows PowerShell. Search, filter, and data aggregation use the GetOpsSummary API action.

Resource data syncs created by a delegated administrator are only available in the delegated administrator account. You can’t view the syncs or the aggregated data in the AWS Organizations master account.

A delegated administrator can create a maximum of five resource data syncs.

A delegated administrator can create a resource data sync for either an entire organization in AWS Organizations or a subset of organizational units.

**Configure an Explorer delegated administrator**

Use the following procedure to register an Explorer delegated administrator.

**To register an Explorer delegated administrator**

1. Log into your AWS Organizations master account.
3. In the navigation pane, choose Explorer.
4. Choose Settings.
5. In the Delegated administrator for Explorer section, verify that you have configured the required service-linked role and service access options. If necessary, choose the buttons to configure these options.

6. For Account ID, enter the AWS account ID. This account must be a member account in AWS Organizations.
7. Choose Register delegated administrator.

The delegated administrator now has access to the following AWS Organizations options on the Create resource data sync page.
Deregister an Explorer delegated administrator

Use the following procedure to deregister an Explorer delegated administrator. A delegated administrator account can only be deregistered by the AWS Organizations master account. When a delegated administrator account is deregistered, the system deletes all AWS Organizations resource data syncs created by the delegated administrator.

To deregister an Explorer delegated administrator

1. Log into your AWS Organizations master account.
3. In the navigation pane, choose Explorer.
4. Choose Settings.
5. In the Delegated administrator for Explorer section, choose Deregister. The system displays a warning.
6. Type the account ID and choose Remove.

The account no longer has access to the AWS Organizations resource data sync API actions. The system deletes all AWS Organizations resource data syncs created by the account.

Using Systems Manager Explorer

This section includes information about how to customize Systems Manager Explorer by changing the widget layout and by changing the data that displays in the dashboard.

Contents

- Editing default rules for OpsItems (p. 131)
- Editing Systems Manager Explorer data sources (p. 132)
- Customizing the display and using filters (p. 133)
- Deleting a Systems Manager Explorer Resource Data Sync (p. 134)

Editing default rules for OpsItems

When you complete Integrated Setup, the system enables more than a dozen rules in Amazon CloudWatch Events. These rules automatically create OpsItems in Systems Manager OpsCenter. Systems Manager Explorer then displays aggregated information about the Opsitems.

Each rule includes a preset Category and Severity value. When the system creates OpsItems from an event, it automatically assigns the preset Category and Severity.

Important

Currently, you can't edit the Category and Severity values for default rules but you can edit these values on OpsItems created from the default rules.
Using Explorer

To edit default rules for creating OpsItems

2. In the navigation pane, choose Explorer.
3. Choose Settings.
4. In the OpsItems rules section, choose Edit.
5. Expand CWE rules.
6. Clear the check box beside those rules that you don't want to use.
7. Use the Category and Severity lists to change this information for a rule.
8. Choose Save.

Your changes take effect the next time the system creates an OpsItem.

Editing Systems Manager Explorer data sources

Systems Manager Explorer displays data from the following sources.

- Amazon Elastic Compute Cloud (Amazon EC2)
- Systems Manager OpsCenter
- Systems Manager Patch Manager
- AWS Trusted Advisor
- AWS Compute Optimizer

Note
You can't configure Explorer to stop displaying OpsCenter OpsItem data.

Before you begin

Verify that you setup and configured services that populate Explorer widgets with data. For more information, see Setting up related services (p. 121).
To edit data sources
2. In the navigation pane, choose Explorer.
3. Choose Settings.
4. In the OpsData sources section, choose Edit.
5. Expand OpsData sources.
6. Add or remove one or more sources.
7. Choose Save.

Customizing the display and using filters

You can customize widget layout in Systems Manager Explorer by using a drag-and-drop capability. You can also customize the OpsData and OpsItems displayed in Explorer by using filters, as described in this topic.

Customizing widget layout

Use the following procedure to customize widget layout in Explorer.

To customize widget layout
2. In the navigation pane, choose Explorer.
3. Choose a widget that you want to move.
4. Click and hold the name of the widget and then drag it to its new location.
5. Repeat this process for each widget that you want to reposition.
If you decide that you don't like the new layout, choose Reset layout to move all widgets back to their original location.

Using filters to the change the data displayed in Explorer

By default, Explorer displays data for the current AWS account and the current Region. If you create one or more resource data syncs, you can use filters to change which sync is active. You can then choose to display data for a specific Region or all Regions. You can also use the Search bar to filter on different OpsItem and key-tag criteria.

To change the data displayed in Explorer by using filters

2. In the navigation pane, choose Explorer.
3. In the Filter section, use the Select a resource data sync list to choose a sync.
4. Use the Regions list to choose either a specific AWS Region or choose All Regions.
5. Choose the Search bar, and then choose the criteria on which to filter the data.
6. Press Enter.

Explorer retains the filter options you selected if you close and reopen the page.

Deleting a Systems Manager Explorer Resource Data Sync

In Systems Manager Explorer, you can aggregate OpsData and OpsItems from other accounts and Regions by creating a resource data sync.

You can't change the account options for a resource data sync. For example, if you created a sync in the us-east-2 (Ohio) Region and you chose the Include only the current account option, you can't edit that sync later and choose the Include all accounts from my AWS Organizations configuration option. Instead, you must delete the resource data sync, and create a new one, as described in the following procedure.

To delete a resource data sync

2. In the navigation pane, choose Explorer.
3. Choose Settings.
4. In the Configure resource data sync section, choose the resource data sync that you want to delete.
5. Choose Delete.

Exporting OpsData from Systems Manager Explorer

When you click a link in Systems Manager Explorer, some pages display OpsData in a list. These pages include an Export button that enables you to export up to 5,000 OpsData items as a comma separated value (.csv) file to an Amazon Simple Storage Service (Amazon S3) bucket. Before you can export data from Explorer pages, you must configure data export as described in this topic.
Before You Begin

When you configure data export, you must specify an Amazon Simple Notification Service (Amazon SNS) topic that exists in the same AWS Region where you want to export the data. Systems Manager sends a notification to the Amazon SNS topic when an export completes. For information about creating an Amazon SNS topic, see Tutorial: Creating an Amazon SNS Topic.

Also, be aware that when you export Explorer data, Systems Manager creates an AWS Identity and Access Management (IAM) role named `AmazonSSMExplorerExportRole`. This role uses the following IAM policy.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [ "s3:PutObject"
         ],
         "Resource": [ "arn:aws:s3:::{ExportDestinationS3BucketName}/*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [ "s3:GetBucketAcl"
         ],
         "Resource": [ "arn:aws:s3:::{ExportDestinationS3BucketName}" ]
      },
      {
         "Effect": "Allow",
         "Action": [ "sns:Publish"
         ],
         "Resource": [ "{{SnsTopicArn}}" ]
      }
   ]
}
```
The role includes the following trust entity.

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

**To export OpsData from Explorer**

2. In the navigation pane, choose Explorer.
3. Choose Settings.
4. In the Configure data export section, choose Edit.
5. To upload the data export file to an existing S3 bucket, choose Select an existing S3 bucket and then choose the bucket from the list. To upload the data export file to a new S3 bucket, choose Create a new S3 bucket, and then enter the name you want to use for the new bucket.
6. Use the Select an Amazon SNS topic ARN list to choose the topic you want to notify when the export completes.
7. Choose Create.
You can now export OpsData from Explorer pages to the specified S3 bucket.

Troubleshooting Systems Manager Explorer

This topic includes information about how to troubleshoot common problems with Systems Manager Explorer.

The AWS Organizations options on the Create resource data sync page are greyed out

The following options on the Create resource data sync page are only available if you set up and configured AWS Organizations. If you set up and configured AWS Organizations, then either the AWS Organizations master account or an Explorer delegated administrator can create resource data syncs that use these options.

For more information, see Setting up Systems Manager Explorer to display data from multiple accounts and Regions (p. 127) and Configuring a Delegated Administrator (p. 129).

Explorer doesn't display any data at all

- Verify that you completed Integrated Setup in each account and Region where you want Explorer to access and display data. If you don't, Explorer won't display OpsData and OpsItems for those accounts and Regions in which you didn't complete Integrated Setup. For more information, see Getting started with Systems Manager Explorer and OpsCenter (p. 121).

- When using Explorer to view data from multiple accounts and Regions, verify that you are logged into the AWS Organizations Master Account. To view OpsData and OpsItems from multiple accounts and Regions, you must be signed in to this account.

Widgets about EC2 instances don't display data

If widgets about EC2 instances, such as the Instance count, Managed instances, and Instance by AMI widgets don't display data, then verify the following:

- Verify that you waited several minutes. OpsData can take several minutes to display in Explorer after you completed Integrated Setup.

- Verify that you configured AWS Config configuration recorder. Explorer uses data provided by AWS Config configuration recorder to populate widgets with information about your EC2 instances. For more information, see Managing the Configuration Recorder.

- Verify that the Amazon EC2 OpsData source is enabled on the Settings page. Also, verify that more than 6 hours have passed since you enabled configuration recorder or since you made changes to your instances. Systems Manager can take up to six hours to display data from AWS Config in Explorer EC2 widgets after you initially enable configuration recorder or make changes to your instances.

- Be aware that if an instance is either stopped or terminated, then Explorer stops showing those instances after 24 hours.

- Verify that you are in the correct AWS Region where you configured your EC2 instances. Explorer doesn't display data about on-premises instances.

- If you configured a resource data sync for multiple accounts and Regions, verify that you are signed in to the Organizations master account.
AWS Systems Manager User Guide
OpsCenter

**Patch widget doesn't display data**

The **Non-compliant instances for patching** widget only displays data about patch instances that are not compliant. This widget displays no data if your instances are compliant. If you suspect that you have non-compliant instances, then verify that you set up and configured Systems Manager patching and use Patch Manager to check your patch compliance. For more information, see AWS Systems Manager Patch Manager (p. 911).

**Miscellaneous issues**

**Explorer doesn't let you edit or remediate OpsItems:** OpsItems viewed across accounts or Regions are read-only. They can only be updated and remediated from their home account or Region.

---

**AWS Systems Manager OpsCenter**

OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. OpsCenter is designed to reduce mean time to resolution for issues impacting AWS resources. This Systems Manager capability aggregates and standardizes OpsItems across services while providing contextual investigation data about each OpsItem, related OpsItems, and related resources. OpsCenter also provides Systems Manager Automation documents (runbooks) that you can use to quickly resolve issues. You can specify searchable, custom data for each OpsItem. You can also view automatically-generated summary reports about OpsItems by status and source.

OpsCenter is integrated with Amazon CloudWatch Events. This means you can create CloudWatch Events rules that automatically create OpsItems for any AWS service that publishes events to CloudWatch Events. For example, you can configure **SSM OpsItems** as the target for the following types of events, and hundreds more:

- Security issues, such as alerts from AWS Security Hub
- Performance issues, such as a throttling event for Amazon DynamoDB or degraded Amazon Elastic Block Store (EBS) volume performance
- Failures, such as an Amazon EC2 Auto Scaling group failure to launch an instance or a Systems Manager Automation execution failure
- Health alerts, such as an AWS Health alert for scheduled maintenance
- State changes, such as an EC2 instance state change from Running to Stopped

OpsCenter is also integrated with Amazon CloudWatch Application Insights for .NET and SQL Server. This means you can automatically create OpsItems for problems detected in your applications.

Operations engineers and IT professionals can create, view, and edit OpsItems by using the OpsCenter page in the AWS Systems Manager console, public API actions, the AWS CLI, AWS Tools for Windows PowerShell, or the AWS SDKs. You can also use AWS Lambda with Amazon SNS to create OpsItems from sources like CloudWatch alarms. OpsCenter public API actions also enable you to integrate OpsCenter with your case management systems and health dashboards.

**How can OpsCenter benefit my organization?**

AWS Systems Manager OpsCenter enables a standard and unified experience for viewing, working on, and remediating issues related to AWS resources. A standard and unified experience improves the time it takes to remedy issues, investigate related issues, and train new operations engineers and IT professionals. A standard and unified experience also reduces the number of manual errors entered into the system of managing and remediating issues.
More specifically, OpsCenter offers the following benefits for operations engineers and organizations:

- You no longer need to navigate across multiple console pages to view, investigate, and resolve OpsItems related to AWS resources. OpsItems are aggregated, across services, in a central location.
- You can view service-specific and contextually relevant data for OpsItems that are automatically generated by Amazon CloudWatch Events and CloudWatch Application Insights for .NET and SQL Server.
- You can specify the Amazon Resource Name (ARN) of a resource related to an OpsItem. By specifying related resources, OpsCenter uses built-in logic to help you avoid creating duplicate OpsItems.
- You can view details and resolution information about similar OpsItems.
- You can quickly view information about and execute Systems Manager Automation documents (runbooks) to resolve issues.

What are the features of OpsCenter?

- **Automated and manual OpsItem creation**
  OpsCenter is integrated with Amazon CloudWatch Events. This means you can create CloudWatch rules that automatically create OpsItems for any AWS service that publishes events to CloudWatch Events. You can also manually create OpsItems.

  OpsCenter is also integrated with Amazon CloudWatch Application Insights for .NET and SQL Server. This means you can automatically create OpsItems for problems detected in your applications.

- **Detailed and searchable OpsItems**
  Each OpsItem includes multiple fields of information, including a title, ID, priority, description, the source of the OpsItem, and the date/time it was last updated. Each OpsItem also includes the following configurable features:
  - **Status**: Open, In progress, Resolved, or Open and In progress.
  - **Related resources**: A related resource is the impacted resource or the resource that triggered the Amazon CloudWatch Events event that created the OpsItem. Each OpsItem includes a Related resources section where OpsCenter automatically lists the Amazon Resource Name (ARN) of the related resource. You can also manually specify ARNs of related resources. For some ARN types, OpsCenter automatically creates a deep link that displays details about the resource without having to visit other console pages to view that information. For example, if you specify the ARN of an EC2 instance, you can view all of the EC2-provided details about that instance in OpsCenter. You can manually add the ARNs of additional related resources. Each OpsItem can list a maximum of 100 related resource ARNs. For more information, see Working with related resources (p. 153).
  - **Related and Similar OpsItems**: The Related OpsItems feature lets you specify the IDs of OpsItems that are in some way related to the current OpsItem. The Similar OpsItem feature automatically reviews OpsItem titles and descriptions and then lists other OpsItems that may be related or of interest to you.
  - **Searchable and private operational data**: Operational data is custom data that provides useful reference details about the OpsItem. For example, you can specify log files, error strings, license keys, troubleshooting tips, or other relevant data. You enter operational data as key-value pairs. The key has a maximum length of 128 characters. The value has a maximum size of 20 KB.

  This custom data is searchable, but with restrictions. For the Searchable operational data feature, all users with access to the OpsItem Overview page (as provided by the DescribeOpsItems API action) can view and search on the specified data. For the Private operational data feature, the data is only viewable by users who have access to the OpsItem (as provided by the GetOpsItem API action).
  - **Deduplication**: By specifying related resources, OpsCenter uses built-in logic to help you avoid creating duplicate OpsItems. Additionally, OpsItems that are automatically created from an event in...
How does OpsCenter work with Amazon CloudWatch Events? Which service should I use?

Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams. Generally speaking, CloudWatch Events lets you know there is a problem with your resources.

OpsCenter helps you investigate and remediate the problem. OpsCenter brings together data from CloudWatch Events or data entered manually by engineers so that your engineers can perform a
though investigation. OpsCenter also provides Automation runbooks for quickly remediating those issues. OpsCenter integrates with CloudWatch Events by enabling you to automatically create OpsItems (or you can manually create OpsItems) to address the following types of issues: performance degradation, state changes, execution failures, maintenance notifications, and security alerts.

**Does OpsCenter integrate with my existing case management system?**

OpsCenter is designed to complement your existing case management systems. You can integrate OpsItems into your existing case management system by using public API actions. You can also maintain manual lifecycle workflows in your current systems and use OpsCenter as an investigation and remediation hub.

For information about OpsCenter public API actions, see the following API actions in the *AWS Systems Manager API Reference*.

- CreateOpsItem
- DescribeOpsItems
- GetOpsItem
- GetOpsSummary
- UpdateOpsItem

**Is there a charge to use OpsCenter?**

Yes. For more information, see *AWS Systems Manager Pricing.*

**Does OpsCenter work with my on-premises and hybrid managed instances?**

Yes. You can use OpsCenter to investigate and remediate issues with your on-premises managed instances that are configured for Systems Manager. For more information about setting up and configuring on-premises servers and virtual machines for Systems Manager, see Setting up AWS Systems Manager for hybrid environments (p. 43).

**Is OpsCenter available in all AWS Regions?**

No. OpsCenter is not available in the BAH (me-south-1) and HKG (ap-east-1) Regions.

**What are the resource limits for OpsCenter?**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Default limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of OpsItems allowed per account per AWS Region (including Open and Resolved OpsItems)</td>
<td>500,000</td>
</tr>
<tr>
<td>Maximum number of OpsItems per account per month</td>
<td>10,000</td>
</tr>
<tr>
<td>Maximum operational data value size</td>
<td>20 KB</td>
</tr>
<tr>
<td>Resource</td>
<td>Default limit</td>
</tr>
<tr>
<td>--------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Maximum number of associated Automation runbooks per OpsItem</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of Automation runbook executions stored in operational data under a single associated runbook</td>
<td>10</td>
</tr>
<tr>
<td>Maximum number of related resources you can specify per OpsItem</td>
<td>100</td>
</tr>
<tr>
<td>Maximum number of related OpsItems you can specify per OpsItem</td>
<td>10</td>
</tr>
<tr>
<td>Maximum length of a deduplication string</td>
<td>64 characters</td>
</tr>
</tbody>
</table>

**Topics**

- Getting started with OpsCenter (p. 142)
- Creating OpsItems (p. 149)
- Working with OpsItems (p. 153)
- Remediating OpsItem issues using Systems Manager Automation (p. 161)
- Viewing OpsCenter summary reports (p. 164)
- Supported resources reference (p. 164)
- Auditing and logging OpsCenter activity (p. 167)

**Getting started with OpsCenter**

Set up for Systems Manager OpsCenter is integrated with set up for Systems Manager Explorer. Explorer is a customizable operations dashboard that reports information about your AWS resources. Explorer displays an aggregated view of operations data (OpsData) for your AWS accounts and across Regions. In Explorer, OpsData includes metadata about your EC2 instances, patch compliance details, and operational work items (OpsItems). Explorer provides context about how OpsItems are distributed across your business units or applications, how they trend over time, and how they vary by category. You can group and filter information in Explorer to focus on items that are relevant to you and that require action. When you identify high priority issues, you can use Systems Manager OpsCenter to run Automation runbooks and quickly resolve those issues.

If you already set up OpsCenter, you still need to complete Integrated Setup to verify settings and options. If you have not set up OpsCenter, then you can use Integrated Setup to get started with both capabilities. For more information, see Getting started with Systems Manager Explorer and OpsCenter (p. 121).

(Optional) Receive OpsItem notifications

You can configure OpsCenter to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic when the systems creates a new OpsItem or updates an existing OpsItem. Complete the following tasks to receive notifications for OpsItems.

- Task 1: Create and subscribe to an Amazon SNS topic (p. 143)
- Task 2: Update the Amazon SNS access policy (p. 143)
- Task 3: Update the AWS KMS access policy (optional) (p. 144)
- Task 4: Enable default OpsItems rules to send notifications for new OpsItems (p. 145)
Task 1: Create and subscribe to an Amazon SNS topic

To receive notifications, you must create and subscribe to an Amazon SNS topic. For more information, see Create a Topic and Subscribing an Endpoint to an Amazon SNS Topic in the Amazon Simple Notification Service Developer Guide.

Note

To receive notifications, you must specify the Amazon Resource Name (ARN) of an Amazon SNS topic that is in the same AWS Region and account as the OpsItem. If you are using OpsCenter in multiple Regions or accounts, then you must create and subscribe to an Amazon SNS topic in each Region or account where you want to receive OpsItem notifications.

Task 2: Update the Amazon SNS access policy

Use the following procedure to update the Amazon SNS access policy so that Systems Manager can publish OpsItem notifications to the Amazon SNS topic you created in task 1.

2. In the navigation pane, choose Topics.
3. Choose the topic you created in task 1, and then choose Edit.

   ▼ Access policy - optional
   This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic. Info

   JSON editor

   ```json
   
   { 
   "Version": "2008-10-17",
   "Id": "__default_policy_ID",
   "Statement": [ 
   { 
   "Sid": "__default_statement_ID",
   "Effect": "Allow",
   "Principal": { 
   "AWS": "*"
   },
   "Action": [ 
   "SNS:GetTopicAttributes",
   "SNS:SetTopicAttributes",
   "SNS:Publish",
   "SNS:AddPermission",
   "SNS:RemovePermission",
   "SNS:DeleteTopic",
   "SNS:ListTopics"
   ],
   "Resource": "arn:aws:sns:AWS_Region:account_ID:topic_name"
   }
   
   5. Add the following Sid block to the existing policy.

   ```json
   
   { 
   "Sid": "Allow OpsCenter to publish to this topic",
   "Effect": "Allow",
   "Principal": { 
   "Service": "ssm.amazonaws.com"
   },
   "Action": [ 
   "SNS:Publish",
   "SNS:RequestNotificationAttributes",
   "SNS:SetNotificationAttributes",
   "SNS:PublishToTopic",
   "SNS:ReceiveNotificationAttributes",
   "SNS:ListNotificationChannels"
   ],
   "Resource": "arn:aws:sns:AWS_Region:account_ID:topic_name"
   }

   ```json

   ```
Enter this block after the existing Sid block. In the following example, the new block is entered at line 29.

```
{ "StringEquals": { 
    "AWS:SourceOwner": "1234567890*
   

```

6. Choose **Save changes**.

The system now sends notifications to the Amazon SNS topic when OpsItems are created or updated.

**Important**
If you configured the Amazon SNS topic with an AWS Key Management Service (AWS KMS) server-side encryption key, then you must complete task 3. If you want to configure OpsItems created by the default OpsItem rules to publish to the Amazon SNS topic, then you must also complete task 4.

**Task 3: Update the AWS KMS access policy (optional)**

If you enabled AWS Key Management Service (AWS KMS) server-side encryption for your Amazon SNS topic, then you must also update the access policy of the AWS KMS customer master key you chose when you configured the topic. Use the following procedure to update the access policy so that Systems Manager can publish OpsItem notifications to the Amazon SNS topic you created in task 1.

**Note**
OpsCenter does not support publishing OpsItems to an Amazon SNS topic configured with an AWS managed key.

2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. In the navigation pane, choose **Customer managed keys**.
4. Choose the ID of the customer master key you chose when you created the topic.
5. In the **Key policy** section, choose **Switch to policy view**.
6. Choose **Edit**.
7. Add the following Sid block to the existing policy.

```json
{
  "Sid": "Allow OpsItems to decrypt the key",
  "Effect": "Allow",
  "Principal": {
    "Service": "ssm.amazonaws.com"
  },
  "Action": ["kms:Decrypt", "kms:GenerateDataKey"],
  "Resource": "arn:aws:kms:AWS_Region:account_ID:key/key_ID"
}
```

Enter this block after one of the existing Sid blocks. In the following example, the new block is entered at line 14.

8. Choose **Save changes**.

**Task 4: Enable default OpsItems rules to send notifications for new OpsItems**

Default OpsItems rules in Amazon CloudWatch Events aren't configured with an ARN for Amazon SNS notifications. Use the following procedure to edit a rule in CloudWatch Events and enter a notifications block.

**To add a notifications block to a default OpsItem rule**

2. In the navigation pane, choose **OpsCenter**.
3. Choose the **OpsItems** tab, and then choose **Configure sources**.
4. Choose the source rule that you want to configure with a notification block, as shown in the following example:
5. On the rule details page, choose **Edit**.

6. Scroll to the **Select targets** section.
7. Enter the notifications block before the resources block, as shown here.

8. Scroll to the bottom of the rule details page and choose Update.

The next time the systems creates an OpsItem for the default rule, it publishes a notification to the Amazon SNS topic.

(Optional) Create OpsItem guidelines for your organization

We recommend that each organization create a simple set of guidelines that promote consistency when creating and editing OpsItems. Guidelines make it easier for users to locate and resolve OpsItems. The guidelines for your organization should define best practices when users enter information into the following OpsItem fields.

**Note**
Amazon CloudWatch Events populates the Title, Source, and Description fields of automatically generated OpsItems. You can edit the Title and the Description fields, but you can't edit the Source field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Guidelines should encourage a consistent OpsItem naming experience. For example, your guidelines might require that each title include information about the impacted resource, the status, the</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>environment, and the name or the alias of the engineer actively working the issue, if applicable. All OpsItems created by CloudWatch include a title that describes the event that caused the creation of the OpsItem, but you can edit these titles. You can search OpsItems for <code>Title:contains</code>. If your naming guidelines encourage consistent use of keywords, you improve your search results.</td>
</tr>
<tr>
<td>Source</td>
<td>Guidelines can include specifying IDs, software version numbers (if applicable) or other relevant data to help users identify the origin of the issue. You can't edit the <code>Source</code> field after the OpsItem is created.</td>
</tr>
<tr>
<td>Priority</td>
<td>(Optional) Guidelines include determining the highest and lowest priority for your organization, and any service-level agreements based on priority. You can specify priority from 1 to 5.</td>
</tr>
<tr>
<td>Description</td>
<td>Guidelines should suggest how much detail about the issue to include and any steps (if applicable) for reproducing the issue.</td>
</tr>
<tr>
<td>Notifications</td>
<td>Guidelines should suggest which Amazon Simple Notification Service (SNS) topic Amazon Resource Name (ARN) to specify when creating or editing OpsItems. Be aware that SNS notifications are region-specific. This means you must specify an ARN that is in the same AWS Region as the OpsItem.</td>
</tr>
<tr>
<td>Related Resources</td>
<td>Guidelines can include details about which Resources should or shouldn't have an ARN specified. For supported AWS resource types, the ARN creates a deep link to details about the Resource.</td>
</tr>
<tr>
<td>Operational data</td>
<td>You can specify custom data for each OpsItem that provides context about the issue and other relevant data for future reference. You can specify searchable custom data. All users with access to the OpsItem Overview page can search for and view this data. You can also specify private custom data that is only viewable by users who have access to this OpsItem. Guidelines could specify structure and standards for key-value pairs. These key-value pairs can describe operational data and resolution details, leading to improved search results.</td>
</tr>
</tbody>
</table>
Creating OpsItems

You can create OpsItems automatically or manually. When you initially configure OpsCenter by using Integrated Setup, you enable Amazon CloudWatch Events to automatically create OpsItems based on common rules. You can also create OpsItems selectively by configuring SSM OpsItems as the target of specific events in CloudWatch Events. For information about enabling default CloudWatch Events rules for creating OpsItems by using Integrated Setup, see Getting started with Systems Manager Explorer and OpsCenter (p. 121).

This section includes the following topics.

- Configuring CloudWatch Events to automatically create OpsItems for specific events (p. 149)
- Integrating with CloudWatch Application Insights for .NET and SQL Server (p. 150)
- Creating OpsItems manually (p. 150)

Configuring CloudWatch Events to automatically create OpsItems for specific events

Use the following procedure to configure SSM OpsItems as the target of a CloudWatch event. When CloudWatch receives the event, it creates a new OpsItem.

**To configure OpsCenter as a target of a CloudWatch event**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Events**, and then either choose to create a new rule or edit an existing rule.
3. After verifying the details of the rule, choose **Add target**.
4. In the **Select target type** list, choose **SSM OpsItem**.
5. For **Configure input**, verify that **Matched event** is selected.
6. In the permissions section, choose **Create a new role for this specific resource** to create a new role with the required permissions. Or, choose **Use existing role** and choose the IAM role you created that gives CloudWatch permission to create OpsItems in OpsCenter. For more information about the required role and permissions, see Getting started with OpsCenter (p. 142).
7. Choose **Configure details**. CloudWatch opens the **Step 2: Configure rule details** page.
8. For **Name**, type a descriptive name that identifies the purpose of this rule.
9. For **Description**, type information about this rule.
10. Verify that the **Enabled** option is selected, and then choose **Create rule**.
11. In the navigation pane, choose **Rules**. Verify that the list includes a new rule that begins with AwsSSMOpsCenter-*, such as AwsSSMOpsCenter-EC2 or AwsSSMOpsCenter-DynamoDB.

After an OpsItem is created from an event, you can view the event details by opening the OpsItem and scrolling down to the **Private operational data** section.

For information about how to configure the options in an OpsItem, see Working with OpsItems (p. 153).
Integrating with CloudWatch Application Insights for .NET and SQL Server

OpsCenter integrates with Amazon CloudWatch Application Insights for .NET and SQL Server. This means you can automatically create OpsItems for problems detected in your applications. For information about how to configure Application Insights to create OpsItems, see Setting Up Your Application in the Amazon CloudWatch User Guide.

Creating OpsItems manually

This section includes procedures for manually create OpsItems for issues that aren't automatically created by Amazon CloudWatch Events.

Before You Begin

If you manually create an OpsItem for an impacted AWS resource, then collect information about that resource so that you can create an Amazon Resource Name (ARN). If you specify an ARN when you create an OpsItem, then OpsCenter automatically creates a deep link to detailed information about the resource. For example, if you specify the ARN of an impacted EC2 instance, then OpsCenter creates a deep link to the details about that instance. For information about how to create an ARN, see the Amazon Resource Names (ARNs) and AWS Service Namespaces in the Amazon Web Services General Reference.

*Note*

OpsCenter does not support creating deep links for all ARN types. To view a list of resources the support deep links based on ARNs, see Supported resources reference (p. 164).

This section includes the following procedures.

- To manually create an OpsItem (console) (p. 150)
- To manually create an OpsItem (AWS CLI) (p. 151)

To manually create an OpsItem (console)

2. In the navigation pane, choose **OpsCenter**.
3. Choose **Create OpsItem**. If you don't see this button, then choose the **OpsItems** tab, and then choose **Create OpsItem**.
4. For **Title**, enter a descriptive name to help you understand the purpose of the OpsItem.
5. For **Source**, enter the type of impacted AWS resource or other source information to help users understand the origin of the OpsItem.

*Note*

You can't edit the **Source** field after you create the OpsItem.
6. For **Priority**, choose the priority level.
7. For **Description**, enter information about this OpsItem including (if applicable) steps for reproducing the issue.
8. For **Deduplication string**, enter words the system should use to check for duplicate OpsItems. For more information about deduplication strings, see Reducing duplicate OpsItems (p. 158).
9. (Optional) For **Notifications**, specify the SNS topic ARN where you want notifications sent when this OpsItem is updated. You must specify an Amazon SNS ARN that is in the same AWS Region as the OpsItem.
10. (Optional) Under **Related resources**, choose **Add** to specify the ARN of the impacted resource and any related resources.
11. Choose **Create OpsItem**.
If successful, the OpsItem opens. For information about how to configure the options in an OpsItem, see Working with OpsItems (p. 153).

**To manually create an OpsItem (AWS CLI)**

1. Open the AWS CLI and run the following command to create an OpsItem.

   ```
   aws ssm create-ops-item --title "Descriptive_title" --description "Information_about_the_issue" --priority Number_between_1_and_5 --source Source_of_the_Issue --operational-data Up_to_20_KB_of_data_or_path_to_JSON_file --notifications Arn="ARN_of_SNS_topic" --tags "Key=key_name,Value=a_value"
   ```

   Here are some examples.

   **Linux**

   ```
   aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 2 --source ec2 --operational-data '{"EC2":{"Value":"12345","Type":"SearchableString"}}' --notifications Arn="arn:aws:sns:us-west-1:12345678:TestUser" --tags "Key=EC2,Value=ProductionServers"
   ```

   The following command uses the /aws/resources key in OperationalData to create an OpsItem with an Amazon DynamoDB related resource.

   ```
   aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 2 --source ec2 --operational-data '{"/aws/resources":{"Value":"
   ```

   The following command uses the /aws/automations key in OperationalData to create an OpsItem that specifies the AWS-ASGEnterStandby document as an associated automation runbook.

   ```
   aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 2 --source ec2 --operational-data '{"/aws/automations":{"Value":"
   [{"automationId": "AWS-ASGEnterStandby", "automationType": "AWS::SSM::Automation"}],"Type":"SearchableString"}}' --notifications Arn="arn:aws:sns:us-west-2:12345678:TestUser"
   ```

   **Windows**

   ```
   aws ssm create-ops-item --title "RDS instance not responding" --description "RDS instance not responding to ping" --priority 1 --source RDS --operational-data={"RDS":{"Value":"abcd","Type":"SearchableString"}} --notifications Arn="arn:aws:sns:us-west-1:12345678:TestUser1" --tags "Key=RDS,Value=ProductionServers"
   ```

   The following command uses the /aws/resources key in OperationalData to create an OpsItem with an EC2 instance related resource.

   ```
   aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 2 --source ec2 --operational-data={"/aws/resources":{"Value":"
   [{"arn": "arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0"}],"Type":"SearchableString"}}
   ```
The following command uses the /aws/automations key in OperationalData to create an OpsItem that specifies the AWS-RestartEC2Instance document as an associated automation runbook.

```bash
aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 2 --source ec2 --operational-data="{"/aws/automations":{"Value":"
[ {
 "automationId": "AWS-RestartEC2Instance",
 "automationType": "AWS::SSM::Automation"
}
]",
"Type":"SearchableString"}"
```

**Specify operational data from a file**

When you create an OpsItem, you can specify operational data from a file. The file must be a JSON file, and the contents of the file must use the following format:

```json
{
 "key_name": {
   "Type": "SearchableString",
   "Value": "Up to 20 KB of data"
 }
}
```

Here is an example.

```bash
aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 2 --source ec2 --operational-data file:///Users/TestUser1/Desktop/OpsItems/opsData.json --notifications Arn="arn:aws:sns:us-west-1:12345678:TestUser1" --tags "Key=EC2,Value=Production"
```

**Note**

For information about how to enter JSON-formatted parameters on the command line on different local operating systems, see [Using quotation marks with strings](https://docs.aws.amazon.com/cli/latest/userguide/cli-aws-usage-on-windows.html) in the *AWS Command Line Interface User Guide*.

The system returns information like the following:

```json
{
 "OpsItemId": "oi-1a2b3c4d5e6f"
}
```

2. Run the following command to view details about the OpsItem you created.

```bash
aws ssm get-ops-item --ops-item-id ID
```

The system returns information like the following:

```json
{
 "OpsItem": {
   "CreatedBy": "arn:aws:iam::12345678:user/TestUser",
   "CreatedTime": 1558386334.995,
   "Description": "Log clean up may have failed which caused the disk to be full",
   "LastModifiedBy": "arn:aws:iam::12345678:user/TestUser",
   "LastModifiedTime": 1558386334.995,
   "Notifications": [
     {
       "Arn": "arn:aws:sns:us-west-1:12345678:TestUser"
     }
   ],
   "Priority": 2,
   "Source": "ec2",
   "State": "OPEN",
   "Title": "EC2 instance disk full",
   "OperationalData": {
     "/aws/automations": {
       "Value": "[{
         "automationId": "AWS-RestartEC2Instance",
         "automationType": "AWS::SSM::Automation"
       }]
     }
   }
  }
}
```
"Priority": 2,
"RelatedOpsItems": [],
"Status": "Open",
"OpsItemId": "oi-1a2b3c4d5e6f",
"Title": "EC2 instance disk full",
"Source": "ec2",
"OperationalData": {
  "EC2": {
    "Value": "12345",
    "Type": "SearchableString"
  }
}
}

3. Run the following command to update the OpsItem. This command changes the status from Open (the default) to InProgress.

   ```bash
   aws ssm update-ops-item --ops-item-id ID --status InProgress
   ```

   The command has no output.

4. Run the following command again to verify that the status changed to InProgress

   ```bash
   aws ssm get-ops-item --ops-item-id ID
   ```

**Working with OpsItems**

This section describes how to configure the options available in an OpsItem. For information about creating OpsItems, see Creating OpsItems (p. 149).

**Topics**

- Working with related resources (p. 153)
- Editing OpsItem details (p. 154)
- Working with related and similar OpsItems (p. 156)
- Working with operational data (p. 156)
- Reducing duplicate OpsItems (p. 158)

**Working with related resources**

A related resource is the impacted resource (the resource that needs to be investigated or the resource that triggered the Amazon CloudWatch Events event that created the OpsItem). Each OpsItem has a Related resources section. If CloudWatch Events creates the OpsItem, then the system automatically populates the OpsItem with the Amazon Resource Name (ARN) of the resource. You can also manually specify ARNs of related resources. For some ARN types, OpsCenter automatically creates a deep link that displays details about the resource without having to visit other console pages to view that information. For example, you can specify the ARN of an EC2 instance. In OpsCenter, you can then view all of the details that Amazon EC2 provides about that instance. To view a list of resource types that automatically create deep links to related resource, see Supported resources reference (p. 164).

**Note**

You can manually add the ARNs of additional related resources. Each OpsItem can list a maximum of 100 related resource ARNs.
To view and add related resources

2. In the navigation pane, choose OpsCenter.
3. Choose the OpsItems tab.
4. Choose an OpsItem ID.

5. To view information about the impacted resource, choose the Related resources details tab.

This tab displays information about the resource from several AWS services. Expand the Resource details section to view information about this resource as provided by the AWS service that hosts it. You can also toggle through other related resources associated with this OpsItem by using the Related resources list.

6. To add additional related resources, choose the Overview tab.
7. In the Related resources section, choose Add.
8. For Resource ARN, enter the ARN of the related resource, and then choose Add.

Note
If you don't know the ARN of the resource, you can manually create it. For information about how to create an ARN, see the Amazon Resource Names (ARNs) and AWS Service Namespaces in the Amazon Web Services General Reference.

Editing OpsItem details

The OpsItem details section includes information about the OpsItem, including the description, title, source, OpsItem ID, and the status, to name a few.
For OpsItems that were created automatically, Amazon CloudWatch Events populates the Title, Source, and Description fields. You can edit the Title and the Description fields, but you can't edit the Source field.

**About OpsItem Status**

When you edit an OpsItem, you can specify a status. The Status list includes the following options:

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Active in the system, but <em>not</em> being worked on by an engineer.</td>
</tr>
<tr>
<td>In progress</td>
<td>Active in the system and being worked on by an engineer.</td>
</tr>
<tr>
<td>Resolved</td>
<td>Not active in the system, but available in Search and when using the Resolved filter on the OpsItem Overview page. You can edit a resolved OpsItem to change the status to Open or In progress.</td>
</tr>
</tbody>
</table>

You can view reports about OpsItem statuses on the Summary tab. For more information, see Viewing OpsCenter summary reports (p. 164).

**About OpsItem Priority**

When you edit an OpsItem, you can choose a priority for that OpsItem by choosing a value between 1 and 5. We recommend that your organization determine what each priority level means and a corresponding service level agreement for each.

**About the Notifications Field**

When you edit an OpsItem, you can specify the ARN of an SNS topic in the Notifications field. By specifying an ARN, you ensure that all stakeholders receive a notification when the OpsItem is edited, including a status change. You may find it helpful to create different ARNs for notifications about different types of AWS resources or different environments. For more information, see the Amazon Simple Notification Service Developer Guide.
Important
The SNS topic must exist in the same AWS Region as the OpsItem. If they are in different regions, the system returns an error.

To edit OpsItem details
2. In the navigation pane, choose OpsCenter.
3. Choose an OpsItem ID to open the details page.
4. In the OpsItem details section, choose Edit.
5. Edit the details of the OpsItem according to the requirements and guidelines specified by your organization.
6. When you are finished, choose Save.

Working with related and similar OpsItems

The Related and Similar OpsItem features are designed to help you investigate operations issues while providing context about the scope of an issue. In the Related OpsItems section, you can specify a maximum of 10 IDs for other OpsItems that are related to the current OpsItem. OpsItems can be related in different ways, including a parent-child relationship between OpsItems, a root cause, or a duplicate.

The Similar OpsItems feature is a system-generated list of OpsItems that may be related or of interest to you. To generate the list, the system scans the titles and descriptions of all OpsItems and returns OpsItems that use similar words.

To add a related OpsItem from similar OpsItems
2. In the navigation pane, choose OpsCenter.
3. Choose an OpsItem ID to open the details page.
4. In the Related OpsItem section, choose Add.
5. For OpsItem ID, specify an ID.
6. Choose Add.

Working with operational data

Operational data is custom data that provides useful reference details about the OpsItem. For example, you can specify log files, error strings, license keys, troubleshooting tips, or other relevant data. You
enter operational data as key-value pairs. The key has a maximum length of 128 characters. The value has a maximum size of 20 KB. You can enter multiple key-value pairs of operational data.

**Important**
Operational data keys *can't* begin with the following: amazon, aws, amzn, ssm, /amazon, /aws, /amzn, /ssm.

You can choose to make the data searchable by other users in the account or you can restrict search access. Searchable data means that all users with access to the OpsItem Overview page (as provided by the DescribeOpsItems API action) can view and search on the specified data. Operational data that is not searchable is only viewable by users who have access to the OpsItem (as provided by the GetOpsItem API action).

**To add operational data to an OpsItem**

2. In the navigation pane, choose **OpsCenter**.
3. Choose an OpsItem ID to open the details page.
4. Expand either **Operational data**.
5. If no operational data exists for the OpsItem, then choose **Add**. If operational data already exists for the OpsItem, choose **Manage**.
6. For **Key**, specify a word or words to help users understand the purpose of the data. The key can't begin with the following: amazon, aws, amzn, ssm, /amazon, /aws, /amzn, /ssm.
7. For **Value**, specify the data.
8. Choose **Save**.

After you create operational data, you can edit the key and the value, remove the operational data, or add additional key-value pairs by choosing **Manage**.
Note
You can filter OpsItems by using the Operational data operator on the OpsItems page. In the Search box, choose Operational data, and then enter a key-value pair in JSON. You must enter the key-value pair by using the following format:
{"key":"key_name","value":"a_value"}

Reducing duplicate OpsItems

OpsCenter uses a combination of built-in logic and configurable deduplication strings to help avoid creating duplicate OpsItems. Deduplication built-in logic is applied anytime the CreateOpsItem API action is called. When creating the OpsItem, Systems Manager creates and stores a hash based on the deduplication string and the resource that trigged the OpsItem. When a request is made to create a new OpsItem, the system checks the deduplication string of the new request. If a matching hash exists for this deduplication string, then Systems Manager doesn't create a new OpsItem.

Note the following information about OpsCenter and deduplication:

- Deduplication strings are not case sensitive. If the system finds a matching deduplication string in an OpsItem, regardless of casing, the new OpsItem isn't created.
- If the system finds a matching deduplication string in an OpsItem, and that OpsItem has a status of Open, then the new OpsItem isn't created. If a matching deduplication string is found in an OpsItem that has a status of Resolved, then the system creates a new OpsItem.
- If the system finds a matching deduplication string in an OpsItem, but the resources are different, then the system creates the new OpsItem.

Configuring deduplication strings

OpsCenter includes the following options for configuring deduplication strings.

- Edit preconfigured deduplication strings: Each of the OpsItem default CloudWatch Events rules includes a preconfigured deduplication string. You can edit these deduplication strings in CloudWatch Events.
- Manually specify deduplication strings: You can enter a deduplication string by using either the Deduplication string field in the console or the OperationalData parameter when you create a new OpsItem by using either the AWS CLI or AWS Tools for Windows PowerShell.

After the system creates an OpsItem, it populates the Deduplication string field, if a string was specified. Here's an example.
After you create an OpsItem, you can't edit or change the deduplication strings in that OpsItem.

This sections includes the following procedures for configuring deduplication strings.

- Editing a deduplication string in an OpsCenter default CloudWatch Events rule (p. 159)
- Specifying a deduplication string by using the AWS CLI (p. 160)

**Note**
For information about entering deduplication strings when you manually create an OpsItem in the console, see Creating OpsItems manually (p. 150).

**Editing a deduplication string in an OpsCenter default CloudWatch Events rule**

Use the following procedure to specify a deduplication string for a CloudWatch Events rule that targets OpsCenter.

**To edit a deduplication string in an OpsItem default CloudWatch Events rule**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose Rules.
3. Choose a rule, and then choose Actions, Edit.

![CloudWatch Console Screenshot](image)

4. In the Targets section, expand the lower Input transformer field and locate the "operationalData": { "/aws/dedup" JSON entry and the deduplication strings that you want to edit.
The deduplication string entry in CloudWatch Events rules uses the following JSON format.

```
"operationalData": { "/aws/dedup": { "type": "SearchableString", "value": "{"dedupString": "Words the system should use to check for duplicate OpsItems"}" }
```

Here is an example.

```
"operationalData": { "/aws/dedup": { "type": "SearchableString", "value": "{"dedupString": "SSMOpsCenter-EBS-volume-performance-issue"}"
```

5. Edit the deduplications strings, and then choose **Configure details** to finish updating the rule.

**Specifying a deduplication string by using the AWS CLI**

You can specify a deduplication string when you manually create a new OpsItem by using the AWS CLI. You enter the deduplication string by using the `OperationalData` parameter. The parameter syntax uses JSON, as shown here.

```
--operational-data '{"/aws/dedup":{"Value":"{"dedupString": "Words the system should use to check for duplicate OpsItems"}","Type":"SearchableString"}}'
```

Here is an example command that specifies a deduplication string of `disk full`.

**Linux**

```
aws ssm create-ops-item --title "EC2 instance disk full" --description "Log clean up may have failed which caused the disk to be full" --priority 1 --source ec2 --operational-data '{"/aws/dedup":{"Value":"{"dedupString": "disk full"}","Type":"SearchableString"}}' --tags "Key=EC2,Value=ProductionServers" --notifications Arn="arn:aws:sns:us-west-1:12345678:TestUser"
```
Remediating OpsItem issues using Systems Manager Automation

AWS Systems Manager Automation helps you quickly remediate issues with AWS resources identified in your OpsItems. Automation uses predefined SSM Automation documents (runbooks) to remediate commons issues with AWS resources. For example, Automation includes runbooks to perform the following actions:

- Stop, start, restart, and terminate Amazon Relational Database Service (Amazon RDS) and Amazon Elastic Compute Cloud (EC2) instances.
- Create AWS resources such as Amazon Machine Images (AMIs), Amazon Elastic Block Store (Amazon EBS) snapshots, and Amazon DynamoDB backups.
- Configure a resource to use AWS services, including Amazon CloudWatch Events, AWS CloudTrail, and Amazon Simple Storage Service (Amazon S3) bucket logging and versioning.
- Attach an AWS Identity and Access Management (IAM) instance profile to an instance.
- Troubleshoot RDP and SSH connectivity issues for EC2 instances.
- Reset access for an EC2 instance.

Each OpsItem in the AWS Management Console includes a Runbooks section, as shown in the following.

If the OpsItem was automatically created by CloudWatch, then the list of runbooks is automatically filtered to reflect the recommended runbooks for the specific issue. If the OpsItem was created manually or by specifying OpsItem as a target of a CloudWatch event, then the Runbooks section lists all SSM Automation runbooks.

You can view information about a runbook by either choosing the runbook name in the console or by using the Systems Manager Automation document details reference (p. 443).

Using a runbook to remediate an OpsItem issue

When you execute a runbook from an OpsItem, you can run a simple version or you can choose the Advanced configuration option. The Advanced configuration opens the runbook in Systems Manager Automation, which provides several options for executing the runbook.
Before You Begin

Before you execute a runbook to remediate an OpsItem issue, do the following:

- Verify that you have permission to run Systems Manager Automation documents. For more information, see Getting started with Automation (p. 281).

- Collect resource-specific ID information for the runbook that you want to execute. For example, if you want to execute a runbook that restarts an EC2 instance, then you must specify the ID of the instance to restart.

To execute a runbook to remediate an OpsItem issue

2. In the navigation pane, choose OpsCenter.
3. Choose the OpsItem ID to open the details page.
4. Scroll to the Runbooks section.
5. Use the Runbooks Search bar or the numbers in the upper right to find the runbook that you want to execute.
6. Choose a runbook, and then choose Execute.
7. Enter the required information for the runbook, and then choose Execute.
8. In the navigation pane, choose **Automation**, and then choose the **Execution ID** link to view the steps and the status of the execution.

![Execution detail: AWS-RestartEC2Instance](image)

**Execution status**

<table>
<thead>
<tr>
<th>Overall status</th>
<th>All executed steps</th>
<th># Succeeded</th>
<th># Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Progress</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Executed steps** (2)

<table>
<thead>
<tr>
<th>Step ID</th>
<th>Step #</th>
<th>Step name</th>
<th>Action</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>d901507c-24a8-4445-b7ed-4f16a3b9b3b6</td>
<td>1</td>
<td>stopInstances</td>
<td>aws:changeInstanceState</td>
<td>Success</td>
</tr>
<tr>
<td>5a083253-50f1-4b68-b5d5-0a007d66e11</td>
<td>2</td>
<td>startInstances</td>
<td>aws:changeInstanceState</td>
<td>In Progress</td>
</tr>
</tbody>
</table>

**Working with associated runbooks**

After you execute a runbook from an OpsItem, the runbook is automatically associated with the related resource of that OpsItem for future reference and easy execution. Associated runbooks are ranked higher than others in the **Runbooks** list, as shown in the following.

![Runbooks](image)

Associated runbooks are also available in the **Run automation** list in the **Related resources** section, as shown in the following.

![Related resources](image)

Use the following procedure to execute a runbook that has already been associated with a related resource in an OpsItem. For information about adding related resources, see **Working with OpsItems** (p. 153).
To execute a resource-associated runbook to remediate an OpsItem issue
2. In the navigation pane, choose **OpsCenter**.
3. Open the OpsItem for which you want to execute a runbook.
4. In the **Related resources** section, choose the resource on which you want to execute a runbook.
5. Choose **Run automation**, and then choose the associated runbook that you want to execute.
6. Enter the required information for the runbook, and then choose **Execute**.
7. In the navigation pane, choose **Automation**, and then choose the **Execution ID** link to view the steps and the status of the execution.

Viewing OpsCenter summary reports

OpsCenter includes a summary page that automatically displays the following information:

- **OpsItem status summary**: a summary of OpsItems by status (Open and In progress, Open, or In Progress).
- **Sources with most open OpsItems**: a breakdown of the top AWS services that have open OpsItems.
- **OpsItems by source and age**: a count of OpsItems, grouped by source and days since creation.

To view the OpsCenter summary page

2. In the navigation pane, choose **OpsCenter**.
3. On the OpsItems Overview page, choose **Summary**.
4. Under **OpsItems by source and age**, choose the Search bar to filter OpsItems according to **Source**. Use the list to filter according to **Status**.

Supported resources reference

OpsCenter automatically creates a deep link to the primary resource page when you specify the Amazon Resource Name (ARN) for the following types of AWS resources. For example, if you specify the ARN of an EC2 instance in the **Related Resources** field, then OpsCenter creates a deep link to the information about that instance in the Amazon EC2 console. This enables you to view detailed information about your impacted AWS resources without having to leave OpsCenter. For more information about adding related resources, see Working with related resources (p. 153).

**Supported resources**

<table>
<thead>
<tr>
<th>Resource name</th>
<th>ARN format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon CloudFront distribution</td>
<td>arn:aws:cloudfront::account-id:*</td>
</tr>
</tbody>
</table>
## Supported resources reference

<table>
<thead>
<tr>
<th>Resource name</th>
<th>ARN format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon CloudWatch alarm</td>
<td><code>arn:aws:cloudwatch:region:account-id:alarm:alarm-name</code></td>
</tr>
<tr>
<td>AWS CloudTrail trail</td>
<td><code>arn:aws:cloudtrail:region:account-id:trail/trailname</code></td>
</tr>
<tr>
<td>AWS CodeBuild project</td>
<td><code>arn:aws:codebuild:region:account-id:resourcetype/resource</code></td>
</tr>
<tr>
<td>Amazon DynamoDB table</td>
<td><code>arn:aws:dynamodb:region:account-id:table/tablename</code></td>
</tr>
<tr>
<td>Amazon EC2 customer gateway</td>
<td><code>arn:aws:ec2:region:account-id:customer-gateway/cgw-id</code></td>
</tr>
<tr>
<td>Amazon EC2 elastic IP</td>
<td><code>arn:aws:ec2:region:account-id:eip/eipallocation-id</code></td>
</tr>
<tr>
<td>Amazon EC2 dedicated host</td>
<td><code>arn:aws:ec2:region:account-id:dedicated-host/host-id</code></td>
</tr>
<tr>
<td>Amazon EC2 instance</td>
<td><code>arn:aws:ec2:region:account-id:instance/instance-id</code></td>
</tr>
<tr>
<td>Amazon EC2 network access control list (ACL)</td>
<td><code>arn:aws:ec2:region:account-id:network-acl/nac-l-id</code></td>
</tr>
<tr>
<td>Amazon EC2 network interface</td>
<td><code>arn:aws:ec2:region:account-id:network-interface/eni-id</code></td>
</tr>
<tr>
<td>Amazon EC2 security group</td>
<td><code>arn:aws:ec2:region:account-id:security-group/security-group-id</code></td>
</tr>
<tr>
<td>Resource name</td>
<td>ARN format</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Amazon EC2 subnet</td>
<td>arn:aws:ec2::region:account-id:subnet/subnet-id</td>
</tr>
<tr>
<td>Amazon EC2 volume</td>
<td>arn:aws:ec2::region:account-id:volume/volume-id</td>
</tr>
<tr>
<td>Amazon EC2 VPC</td>
<td>arn:aws:ec2::region:account-id:vpc/vpc-id</td>
</tr>
<tr>
<td>Amazon EC2 VPN connection</td>
<td>arn:aws:ec2::region:account-id:vpn-connection/vpn-id</td>
</tr>
<tr>
<td>Amazon EC2 VPN gateway</td>
<td>arn:aws:ec2::region:account-id:vpn-gateway/vgw-id</td>
</tr>
<tr>
<td>AWS Elastic Beanstalk application</td>
<td>arn:aws:elasticbeanstalk::region:account-id:application/applicationname</td>
</tr>
<tr>
<td>Elastic Load Balancing (classic load balancer)</td>
<td>arn:aws:elasticloadbalancing::region:account-id:loadbalancer/name</td>
</tr>
<tr>
<td>Elastic Load Balancing (application load balancer)</td>
<td>arn:aws:elasticloadbalancing::region:account-id:loadbalancer/app/load-balancer-name/load-balancer-id</td>
</tr>
<tr>
<td>AWS Identity and Access Management (IAM) group</td>
<td>arn:aws:iam::account-id:group/group-name</td>
</tr>
<tr>
<td>IAM policy</td>
<td>arn:aws:iam::account-id:policy/policy-name</td>
</tr>
<tr>
<td>IAM role</td>
<td>arn:aws:iam::account-id:role/role-name</td>
</tr>
<tr>
<td>IAM user</td>
<td>arn:aws:iam::account-id:user/user-name</td>
</tr>
<tr>
<td>AWS Lambda function</td>
<td>arn:aws:lambda::region:account-id:function:function-name</td>
</tr>
<tr>
<td>Amazon Relational Database Service (Amazon RDS) cluster</td>
<td>arn:aws:rds::region:account-id:cluster:db-cluster-name</td>
</tr>
</tbody>
</table>
## Auditing and logging OpsCenter activity

AWS CloudTrail captures OpsCenter API calls made in the AWS Systems Manager console, the AWS CLI, and the Systems Manager SDK. You can view the information in the CloudTrail console or in an Amazon Simple Storage Service (Amazon S3) bucket. One bucket is used for all CloudTrail logs for your account.
Logs of OpsCenter actions show create, update, get, and describe OpsItem activities. For more information about viewing and using CloudTrail logs of Systems Manager activity, see Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144).

Amazon CloudWatch dashboards hosted by Systems Manager

Amazon CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources. With dashboards, you can create the following:

- A single view for selected metrics and alarms to help you assess the health of your resources and applications across one or more regions. You can select the color used for each metric on each graph, so that you can easily track the same metric across multiple graphs.
- An operational playbook that provides guidance for team members during operational events about how to respond to specific incidents.
- A common view of critical resource and application measurements that can be shared by team members for faster communication flow during operational events.

You can create dashboards by using the console, the AWS CLI, or by using the PutDashboard API. For more information, see Using Amazon CloudWatch Dashboards in the Amazon CloudWatch User Guide.

Trusted Advisor and Personal Health Dashboards hosted by Systems Manager

Systems Manager hosts two online tools to help you provision your resources and monitor your account for health events. Trusted Advisor is an online tool that provides you real time guidance to help you provision your resources following AWS best practices. For more information, see Trusted Advisor.

The AWS Personal Health Dashboard provides information about AWS Health events that can affect your account. The information is presented in two ways: a dashboard that shows recent and upcoming events organized by category, and a full event log that shows all events from the past 90 days. For more information, see Getting Started with the AWS Personal Health Dashboard.

**Important**

Trusted Advisor and AWS Personal Health Dashboard display data in Systems Manager only in the following AWS Regions.

<table>
<thead>
<tr>
<th>Region name</th>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
</tr>
<tr>
<td>US West (N. California)</td>
<td>us-west-1</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
</tr>
<tr>
<td>Region name</td>
<td>Region</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Asia Pacific (Osaka-Local)</td>
<td>ap-northeast-3</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
</tr>
<tr>
<td>Europe (Frankfurt)</td>
<td>eu-central-1</td>
</tr>
<tr>
<td>Europe (Ireland)</td>
<td>eu-west-1</td>
</tr>
<tr>
<td>Europe (London)</td>
<td>eu-west-2</td>
</tr>
<tr>
<td>Europe (Paris)</td>
<td>eu-west-3</td>
</tr>
<tr>
<td>Europe (Stockholm)</td>
<td>eu-north-1</td>
</tr>
<tr>
<td>South America (São Paulo)</td>
<td>sa-east-1</td>
</tr>
</tbody>
</table>
AWS Systems Manager Application Management

Application Management is a suite of capabilities that help you manage your applications running in AWS.

Topics
- Resource Groups in AWS Systems Manager (p. 170)
- AWS AppConfig (p. 175)
- AWS Systems Manager Parameter Store (p. 198)

Resource Groups in AWS Systems Manager

You can use resource groups to organize your AWS resources. Resource groups make it easier to manage, monitor, and automate tasks on large numbers of resources at one time.

AWS Resource Groups provides two general methods for defining a resource group. Both methods involve using a query to identify the members for a group.

The first method relies on tags applied to AWS resources to add resources to a group. Using this method, you apply the same key/value pair tags to resources of various types in your account and then use the AWS Resource Groups service to create a group based on that tag pair.

The second method is based on resources available in an individual AWS CloudFormation stack. Using this method, you choose an AWS CloudFormation stack, and then choose resource types in the stack that you want to be in the group.

For more information about these methods, see Build Queries and Groups in AWS Resource Groups in the AWS Resource Groups User Guide.

In order to be added to a resource group, the resources must all be in the same AWS Region. The resources must also be one of the resource types supported for use in resource groups. For example, in the AWS Identity and Access Management (IAM) service, you can apply tags to both users and roles, but only roles are supported by resource groups. In the Amazon Simple Notification Service (Amazon SNS), you can add tags to topics to add them to a resource group, but you can't add tags to subscriptions. In AWS Systems Manager, you can apply tags to several resource types, including maintenance windows, managed instances, and patch baselines, but not to change calendars or Distributor packages.

You can add tags to your AWS resources in different ways:
- When you create the resource
- When you update the resource
- Using the Tag Editor in the Resource Groups service

Supported AWS resource types

For a list of all services with resources types that can be added to resource groups through the use of tags, and the resources they support, see Supported Resources in the AWS Resource Groups User Guide.

Supported Systems Manager resource types
The Systems Manager resource types that you can tag in order to add them to resource groups include the following:

- SSM documents
- Managed instances
- Maintenance windows
- SSM parameters
- Patch baselines

Permissions to work with resource groups and Tag Editor

Before users in your AWS account can work with resource groups and tags in the Resource Groups service and Tag Editor, a user with administrator access must provide the users with the necessary permissions. For information about granting the Systems Manager users in your account access to Resource Groups and Tag Editor in the AWS Management Console and the Resource Groups capability in Systems Manager, see Create user groups (p. 27).

Using the Tag Editor

Using the Tag Editor is the most efficient way to add many resource types to a resource group. You can view all supported resource types in your account from the same page. For resources of certain types, choose just the resources you want to add to the resource group, and add the tags to them in bulk. For information about using the Tag Editor, see Find Resources to Tag and Manage Tags in the AWS Resource Groups User Guide.

What else can I use tagged resources for?

Adding resources to a resource group is just one major use of resource tags. You can also use tags to specify resources with certain tags applied as the targets of AWS operations. You can search for resources with the same tags applied to them. You can craft IAM policies to grant or deny users access to resources that are tagged with those tags.

What can I do with resource groups?

Several AWS services, including Systems Manager, let you act on, monitor, or share AWS resources as a group. For example, in Amazon CloudWatch, you can focus your view to display metrics and alarms from a single resource group. In AWS Resource Access Manager, you can share the AWS resources you have added to a resource group, as a group, with other accounts that you choose.

For information about all AWS services that can use resource groups, see Service Integrations with AWS Resource Groups in the AWS Resource Groups User Guide.

In Systems Manager, you can work with resource groups in a number of ways.

First, you can create and manage resource groups. Systems Manager includes a console that provides the same functionality as the Resource Groups service console. To access this console, do the following:

2. In the navigation pane, choose Resource Groups.

For information about working with the Resource Groups console, see Getting Started with AWS Resource Groups in the AWS Resource Groups User Guide.

Second, in the Inventory capability, you can select the managed instances for which you want to view compliance data by specifying a resource group to which they belong.

Third, you can specify a resource group as the target for the following:
• A command you run in Run Command.
• An Automation workflow you run in Automation.
• An association you create in State Manager.
• A maintenance window you create in Maintenance Windows.
• A package installation or update operation in Distributor.

Topics
• Viewing operations data for AWS Resource Groups (p. 172)

Related AWS Blog Posts
Refer to the following AWS blog posts for more information about resource groups.
• Use Tag Policies to Manage Tags Across Multiple AWS Accounts (26 NOV 2019)
• Simplify granting access to your AWS resources by using tags on AWS IAM users and roles (19 NOV 2018)
• The Resource Groups Tagging API Makes It Easier to List Your Resources by Using a New Pagination Parameter (22 MAY 2017)
• Centrally Manage Tags and Search for Resources Across AWS Services by Using the New Resource Groups Tagging API (30 MAR 2017)
• Organize Your AWS Resources by Using up to 50 Tags per Resource (15 AUG 2016, updated 28 DEC 2017)

Viewing operations data for AWS Resource Groups
In the AWS Systems Manager console, the AWS Resource Groups page displays operations data for a selected group on five tabs: Details, Config, CloudTrail, Monitoring, and OpsItems.

Note
These tabs are not available when viewing a group in the Resource Groups console.
You can use the information on these tabs to help you understand which resources in a group are compliant and working correctly and which resources require action. If you need to take action on a resource, you can use Systems Manager Automation runbooks to perform common operations maintenance and troubleshooting tasks.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
<td>For tag-based resource groups, this tab includes an overview of the group, the tags on which the group is based, and a list of the resources discovered by the resource group query. For AWS CloudFormation stack-based groups, this tab includes detailed information about the AWS CloudFormation stack, including stack events. For tag-based and AWS CloudFormation stack-based resource groups, if a resource in the Group resources section includes a link, you can choose this link to view additional details about the selected resource. Not all AWS resources are supported for the additional details view. For more information, see Resources supported in the additional details view (p. 173) in this topic.</td>
</tr>
<tr>
<td>Config</td>
<td>This tab includes AWS Config compliance and history information for resources in the selected group.</td>
</tr>
<tr>
<td>CloudTrail</td>
<td>This tab lists AWS CloudTrail events for resources in the selected group.</td>
</tr>
<tr>
<td>Monitoring</td>
<td>This tab lists Amazon CloudWatch alarms for resources in the selected group. This tab also shows CloudWatch dashboards associated with your resource group. The tab is empty if no alarms have been configured and no dashboards created for the selected resource group.</td>
</tr>
<tr>
<td>OpsItems</td>
<td>This tab lists all operational work items (OpsItems) for AWS CloudFormation stack-based resources in the selected group. OpsItems for tag-based resources are not supported. For tag-based resources, the OpsItems table is empty. The table is also empty if no OpsItems were created for the AWS CloudFormation resources in this group.</td>
</tr>
</tbody>
</table>

**Important**

Before you can view OpsItems on the OpsItems tab, you must set up and configure AWS Systems Manager Explorer. For more information, see Getting Started with Systems Manager Explorer and OpsCenter in the AWS Systems Manager User Guide. After you set up and configure Explorer, Systems Manager automatically populates the OpsItems tab.

**Resources supported in the additional details view**

The Details tab for tag-based and AWS CloudFormation stack-based groups includes a Group resources section. This section lists each resource in the selected group.
For tag-based and AWS CloudFormation stack-based resource groups, if a resource in this section includes a link, you can choose this link to view additional details about the selected resource. The additional details page includes operational data from multiple AWS services to help you quickly assess the status of a resource and recent actions applied to it.

Not all AWS resources are supported for the additional details view. The following list shows the AWS services that are supported. When listed in the **Group resources** section, these resources include a link to the additional details page.

<table>
<thead>
<tr>
<th>Name</th>
<th>Service</th>
<th>Type</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2 instance i-02d64a93286a6f756</td>
<td>EC2</td>
<td>Instance</td>
<td>i-02d64a93286a6f756</td>
</tr>
<tr>
<td>EC2 instance i-06b7615a1d444da00</td>
<td>EC2</td>
<td>Instance</td>
<td>i-06b7615a1d444da00</td>
</tr>
<tr>
<td>EC2 instance i-076345d819f50d59f</td>
<td>EC2</td>
<td>Instance</td>
<td>i-076345d819f50d59f</td>
</tr>
<tr>
<td>EC2 instance i-08efb0992a89fcb9d</td>
<td>EC2</td>
<td>Instance</td>
<td>i-08efb0992a89fcb9d</td>
</tr>
</tbody>
</table>

ACM::Certificate
AutoScaling::AutoScalingGroup
CloudFront::Distribution
CloudFormation::Stack
CloudWatch::Alarm
CloudTrail::Trail
CodeBuild::Project
CodePipeline::Pipeline
DynamoDB::Table
EC2::CustomerGateway
EC2::EIP
EC2::Host
EC2::Instance
EC2::InternetGateway
EC2::NetworkAcl
EC2::NetworkInterface
EC2::RouteTable
EC2::SecurityGroup
EC2::Subnet
EC2::Volume
EC2::VPC
Use AWS AppConfig, a capability of AWS Systems Manager, to create, manage, and quickly deploy application configurations. AppConfig supports controlled deployments to applications of any size and includes built-in validation checks and monitoring. You can use AppConfig with applications hosted on EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices.

To prevent errors when deploying application configurations, especially for production systems where a simple typo could cause an unexpected outage, AppConfig includes validators. A validator provides a syntactic or semantic check to ensure that the configuration you want to deploy works as intended. To validate your application configuration data, you provide a schema or Lambda function that runs against the configuration. The configuration deployment or update can only proceed when the configuration data is valid.

During a configuration deployment, AppConfig monitors the application to ensure that the deployment is successful. If the system encounters an error, AppConfig rolls back the change to minimize impact for your application users. You can configure a deployment strategy for each application or environment that includes deployment criteria, including velocity, bake time, and alarms to monitor. Similar to error monitoring, if a deployment triggers an alarm, AppConfig automatically rolls back to the previous version.

AppConfig supports multiple use cases. Here are some examples.

- **Application tuning**: Use AppConfig to carefully introduce changes to your application that can only be tested with production traffic.
- **Feature toggle**: Use AppConfig to turn on new features that require a timely deployment, such as a product launch or announcement.
- **Allow list**: Use AppConfig to allow premium subscribers to access paid content.
- **Operational issues**: Use AppConfig to reduce stress on your application when a dependency or other external factor impacts the system.
How can AppConfig benefit my organization?

AppConfig offers the following benefits.

• **Deploy changes across a set of targets quickly**

  AppConfig simplifies the administration of applications at scale by deploying configuration changes from a central location. AppConfig supports configurations stored in Systems Manager Parameter Store, Systems Manager (SSM) documents, and Amazon S3. You can use AppConfig with applications hosted on EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices.

• **Reduce errors in configuration changes**

  AppConfig reduces application downtime by enabling you to create rules to validate your configuration. Configurations that aren’t valid can’t be deployed. AppConfig provides two options for validating configurations.
  - For syntactic validation, you can use a JSON schema. AppConfig validates your configuration by using the JSON schema to ensure that configuration changes adhere to the application requirements.
  - For semantic validation, you can call an AWS Lambda function that runs your configuration before you deploy it.

• **Update applications without interruptions**

  AppConfig deploys configuration changes to your targets at runtime without a heavy-weight build process or taking your targets out of service.

• **Control deployment of changes across your application**

  When deploying configuration changes to your targets, AppConfig enables you to minimize risk by using a deployment strategy. You can use the rate controls of a deployment strategy to determine how fast you want your application targets to receive a configuration change.

What types of targets are supported?

You can use AppConfig with applications hosted on EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices. Targets don’t need to be configured with the Systems Manager SSM Agent or the AWS Identity and Access Management (IAM) instance profile required by other Systems Manager capabilities. This means that AppConfig works with unmanaged instances.

Is there a charge to use AppConfig?

Yes. For more information, see [AWS Systems Manager Pricing](https://aws.amazon.com/sms-pricing/).

Do I have to change my application to work with AppConfig?

Yes. You must configure your application to poll for new configuration updates by using the `GetConfiguration` API action. When a new or updated configuration is ready, AppConfig deploys the configuration file to each target in your deployment strategy.

How does AppConfig work?

At a high level, there are three processes for working with AppConfig.
1. Configure AppConfig to work with your application.
2. Enable your application code to periodically check for and receive configuration data from AppConfig.
3. Deploy a new or updated configuration.

**Configure AppConfig to work with your application**

To configure AppConfig to work with your application, you set up three types of resources.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>An application in AppConfig is a logical unit of code that provides capabilities for your customers. For example, an application can be a microservice that runs on EC2 instances, a mobile application installed by your users, a serverless application using Amazon API Gateway and AWS Lambda, or any system you run on behalf of others.</td>
</tr>
<tr>
<td>Environment</td>
<td>For each application, you define one or more environments. An environment is a logical deployment group of AppConfig applications, such as applications in a Beta or Production environment. You can also define environments for application subcomponents such as the Web, Mobile, and Back-end components for your application. You can configure Amazon CloudWatch alarms for each environment. The system monitors alarms during a configuration deployment. If an alarm is triggered, the system rolls back the configuration.</td>
</tr>
<tr>
<td>Configuration profile</td>
<td>A configuration profile includes source information for accessing your configuration data in a Systems Manager (SSM) document, a Parameter Store parameter, or in Amazon S3. A configuration profile can also include optional validators to ensure your configuration data is syntactically and semantically correct. AppConfig performs a check using the validators when you start a deployment. If any errors are detected, the deployment stops before making any changes to the targets of the configuration.</td>
</tr>
</tbody>
</table>

**Enable your application code to periodically check for and receive configuration data from AppConfig**

Using the `GetConfiguration` API action, AppConfig offers dynamic configuration updates in a controlled manner. You must configure your application code to call this API action on a periodic basis. When called, your code sends the following information:

- The IDs of an AppConfig application, environment, and configuration profile.
- A unique application instance identifier called a client ID.
- The last configuration version known by your application code.
When a new configuration is deployed (which means there is a new version of the configuration), AppConfig responds to the GetConfiguration request and returns the new configuration data.

**Deploy a new or updated configuration**

AppConfig enables you to deploy configurations in the manner that best suits the use case of your applications. You can deploy changes in seconds or you can roll them out slowly to assess the impact of the changes. The AppConfig resource that helps you control deployments is called a deployment strategy. A deployment strategy includes the following information:

- Total amount of time for a deployment to last. \(\text{DeploymentDurationInMinutes}\).
- The percentage of targets to receive a deployed configuration during each interval. \(\text{GrowthFactor}\).
- The amount of time AppConfig monitors for alarms before considering the deployment to be complete and no longer eligible for automatic rollback. \(\text{FinalBakeTimeInMinutes}\).

You can use built-in deployment strategies that cover common scenarios or you can create your own. After you create or choose a deployment strategy, you start the deployment. Starting the deployment calls the StartDeployment API action. The call includes the IDs of an application, environment, configuration profile, and (optionally) the configuration data version to deploy. The call also includes the ID of the deployment strategy to use, which determines how the configuration data rolls out.

**What are the Service Quotas of AppConfig?**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppConfig applications</td>
<td>100</td>
</tr>
<tr>
<td>Deployment strategies</td>
<td>20</td>
</tr>
<tr>
<td>Environments per AppConfig application</td>
<td>20</td>
</tr>
<tr>
<td>Configuration profiles per AppConfig application</td>
<td>100</td>
</tr>
</tbody>
</table>

For information about getting started with and using AppConfig, see the following topics.

**Topics**

- Getting started with AWS AppConfig (p. 178)
- Working with AWS AppConfig (p. 182)

**Getting started with AWS AppConfig**

The following topics describe important tasks for getting started with AWS AppConfig.

**Topics**

- Configuring permissions for AWS AppConfig (p. 178)
- (Optional) Configuring permissions for rollback based on CloudWatch alarms (p. 180)

**Configuring permissions for AWS AppConfig**

AWS AppConfig uses the following API actions.
<table>
<thead>
<tr>
<th>AppConfig resource</th>
<th>API actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppConfig applications</td>
<td>CreateApplication, UpdateApplication, DeleteApplication, GetApplication,</td>
</tr>
<tr>
<td></td>
<td>ListApplications</td>
</tr>
<tr>
<td></td>
<td>For more information, see Step 1: Creating an AppConfig application (p. 183)</td>
</tr>
<tr>
<td>Environments</td>
<td>CreateEnvironment, UpdateEnvironment, DeleteEnvironment, GetEnvironment,</td>
</tr>
<tr>
<td></td>
<td>ListEnvironments</td>
</tr>
<tr>
<td></td>
<td>For more information, see Step 2: Creating an environment (p. 183)</td>
</tr>
<tr>
<td>Configurations</td>
<td>GetConfiguration</td>
</tr>
<tr>
<td></td>
<td>For more information, see Step 3: Creating a configuration and a configuration profile (p. 184)</td>
</tr>
<tr>
<td>Configuration profiles</td>
<td>CreateConfigurationProfile, UpdateConfigurationProfile, DeleteConfigurationProfile, GetConfigurationProfile, ValidateConfiguration, ListConfigurationProfiles, ValidateConfiguration</td>
</tr>
<tr>
<td></td>
<td>For more information, see Step 3: Creating a configuration and a configuration profile (p. 184)</td>
</tr>
<tr>
<td>Deployment strategies</td>
<td>CreateDeploymentStrategy, UpdateDeploymentStrategy, DeleteDeploymentStrategy,</td>
</tr>
<tr>
<td></td>
<td>For more information, see Step 4: Creating a deployment strategy (p. 194)</td>
</tr>
<tr>
<td>Deployments</td>
<td>StartDeployment, StopDeployment, ListDeployments</td>
</tr>
<tr>
<td></td>
<td>For more information, see Step 5: Deploying a configuration (p. 196)</td>
</tr>
</tbody>
</table>

We recommend that you create restrictive IAM permissions policies that grant users, groups, and roles the least privileges necessary to perform a desired action in AppConfig.

For example, you can create a read-only IAM permissions policy that includes only the `Get` and `List` API actions used by AppConfig, like the following.

```json
{
 "Version": "2012-10-17",
 "Statement": [
```
Important
Restrict access to the StartDeployment and StopDeployment API actions to trusted users who understand the responsibilities and consequences of deploying a new configuration to your targets.

For more information about creating and editing IAM policies, see Creating IAM Policies in the IAM User Guide. For information about how to assign this policy to an IAM group, see Attaching a Policy to an IAM Group.

To configure an IAM user account with permission to use AppConfig

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users.
3. In the list, choose a name.
4. Choose the Permissions tab.
5. On the right side of the page, under Permission policies, choose Add inline policy.
6. Choose the JSON tab.
7. Replace the default content with your custom permissions policy.
9. On the Review policy page, for Name, enter a name for the inline policy. For example: AppConfig-<action>-Access.
10. Choose Create policy.

(Optional) Configuring permissions for rollback based on CloudWatch alarms

You can configure AWS AppConfig to roll back to a previous version of a configuration in response to one or more Amazon CloudWatch alarms. When you configure a deployment to respond to CloudWatch alarms, you specify an AWS Identity and Access Management (IAM) role. AppConfig requires this role so that it can monitor CloudWatch alarms even if those alarms weren't created in the current AWS account.

Use the following procedures to create an IAM role that enables AppConfig to rollback based on CloudWatch alarms. This section includes the following procedures.
1. Step 1: Create the permission policy for rollback based on CloudWatch alarms (p. 181)
2. Step 2: Create the IAM role for rollback based on CloudWatch alarms (p. 181)
3. Step 3: Add a trust relationship (p. 182)

Step 1: Create the permission policy for rollback based on CloudWatch alarms

Use the following procedure to create an IAM policy that gives AppConfig permission to call the DescribeAlarms API action.

To create an IAM permission policy for rollback based on CloudWatch alarms

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy.
3. On the Create policy page, choose the JSON tab.
4. Replace the default content on the JSON tab with the following permission policy, and then choose Review.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "cloudwatch:DescribeAlarms"
         ],
         "Resource": "*
      }
   ]
}
```
5. On the Review page, enter SSMCloudWatchAlarmDiscoveryRole in the Role name field.
6. Choose Create policy. The system returns you to the Policies page.

Step 2: Create the IAM role for rollback based on CloudWatch alarms

Use the following procedure to create an IAM role and assign the policy you created in the previous procedure to it.

To create an IAM role for rollback based on CloudWatch alarms

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
3. Under Select type of trusted entity, choose AWS service.
4. Immediately under Choose the service that will use this role, choose EC2, and then choose Next: Permissions.
5. On the Attached permissions policy page, search for SSMCloudWatchAlarmDiscoveryRole.
6. Choose this policy and then choose Next: Tags.
7. Enter tags for this role, and then choose Next: Review.
8. On the Create role page, enter a name in the Role name field, and then choose Create role.

**Step 3: Add a trust relationship**

Use the following procedure to configure the role you just created to trust AppConfig.

**To add a trust relationship for AppConfig**

1. In the Summary page for the role you just created, choose the Trust Relationships tab, and then choose Edit Trust Relationship.
2. Edit the policy to include only "appconfig.amazonaws.com", as shown in the following example:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "appconfig.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```
3. Choose Update Trust Policy.

**Working with AWS AppConfig**

This section includes topics that describe how to use AWS AppConfig features to create and deploy a configuration to your hosts or targets.

**Topics**

- Step 1: Creating an AppConfig application (p. 183)
- Step 2: Creating an environment (p. 183)
- Step 3: Creating a configuration and a configuration profile (p. 184)
- Step 4: Creating a deployment strategy (p. 194)
Step 1: Creating an AppConfig application

An application in AWS AppConfig is a logical unit of code that provides capabilities for your customers. For example, an application can be a microservice that runs on EC2 instances, a mobile application installed by your users, a serverless application using Amazon API Gateway and AWS Lambda, or any system you run on behalf of others.

Use the following procedure to create an AppConfig application by using the AWS Systems Manager console.

To create an application

2. In the navigation pane choose AppConfig.
3. On the Applications tab, choose Create application.
4. For Name, enter a name for the application.
5. For Description, enter information about the application.
6. In the Tags section, enter a key and an optional value. You can specify a maximum of 50 tags for a resource.
7. Choose Create application.

AppConfig creates the application and then displays the Environments tab. Proceed to Step 2: Creating an environment (p. 183). You can begin the procedure where it states, "On the Environments tab..."

Step 2: Creating an environment

For each AWS AppConfig application, you define one or more environments. An environment is a logical deployment group of AppConfig targets, such as applications in a Beta or Production environment. You can also define environments for application subcomponents such as the Web, Mobile, and Back-end components for your application. You can configure Amazon CloudWatch alarms for each environment. The system monitors alarms during a configuration deployment. If an alarm is triggered, the system rolls back the configuration.

Before You Begin

If you want to enable AppConfig to roll back a configuration in response to a CloudWatch alarm, then you must configure an AWS Identity and Access Management (IAM) role with permissions to enable AppConfig to respond to CloudWatch alarms. You choose this role in the following procedure. For more information, see (Optional) Configuring permissions for rollback based on CloudWatch alarms (p. 180).

Use the following procedure to create an AppConfig environment by using the AWS Systems Manager console.

To create an environment

2. In the navigation pane choose AppConfig.
3. On the Applications tab, choose the application you created in Step 1: Creating an AppConfig application (p. 183) and then choose View details.
4. On the Environments tab, choose Create environment.
5. For **Name**, enter a name for the environment.
6. For **Description**, enter information about the environment.
7. In the **Monitors** section, choose **Enable rollback on CloudWatch Alarms** if you want AppConfig to roll back a configuration when an alarm is triggered.
8. In the **IAM role** list, choose the IAM role with permission to roll back a configuration when an alarm is triggered.
9. In the **CloudWatch alarms** list, choose one or more alarms to monitor.
10. In the **Tags** section, enter a key and an optional value. You can specify a maximum of 50 tags for a resource.
11. Choose **Create environment**.

AppConfig creates the environment and then displays the **Environment details** page. Proceed to **Step 3: Creating a configuration and a configuration profile (p. 184)**.

### Step 3: Creating a configuration and a configuration profile

A **configuration** is a collection of settings that influence the behavior of your application. For example, you can create and deploy configurations that carefully introduce changes to your application or turn on new features that require a timely deployment, such as a product launch or announcement. Here's a very simple example of an access list configuration:

```json
{
   "AccessList": [
   {
   "user_name": "Mateo_Jackson"
   },
   {
   "user_name": "Jane_Doe"
   }
   ]
}
```

A **configuration profile** enables AWS AppConfig to access your configuration in its stored location. You can store configurations in one of the following locations:

- Objects in an Amazon Simple Storage Service (Amazon S3) bucket
- Documents in the Systems Manager document store
- Parameters in Parameter Store

A configuration profile includes the following information:

- The URI location where the configuration is stored.
- The AWS Identity and Access Management (IAM) role that provides access to the configuration.
- A validator for the configuration data. You can use either a JSON Schema or an AWS Lambda function to validate your configuration profile. A configuration profile can have a maximum of two validators.

For configurations stored in SSM documents, you can create the configuration by using the Systems Manager console at the time you create a configuration profile. The process is described later in this topic.

For configurations stored in SSM parameters or in S3, you must create the parameter or object first and then add it to Parameter Store or S3. After you create the parameter or object, you can use the
procedure in this topic to create the configuration profile. For information about creating a parameter in Parameter Store, see Creating Systems Manager parameters (p. 249).

Topics
- About configurations stored in Amazon S3 (p. 185)
- About configuration store quotas and limitations (p. 188)
- About validators (p. 189)
- About the configuration profile IAM role (p. 190)
- Creating a configuration and a configuration profile (p. 191)

About configurations stored in Amazon S3

You can store configurations in an Amazon Simple Storage Service (Amazon S3) bucket. When you create the configuration profile, you specify the URI to a single S3 object in a bucket. You also specify the Amazon Resource Name (ARN) of an AWS Identity and Access Management (IAM) role that gives AppConfig permission to get the object. Before you create a configuration profile for an Amazon S3 object, be aware of the following restrictions.

<table>
<thead>
<tr>
<th>Restriction</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Configurations stored as S3 objects can be a maximum of 1 MB in size.</td>
</tr>
<tr>
<td>Object encryption</td>
<td>A configuration profile can't target an encrypted S3 object.</td>
</tr>
<tr>
<td>Storage classes</td>
<td>AppConfig supports the following S3 storage classes: STANDARD, INTELLIGENT_TIERING, REDUCED_REDUNDANCY, STANDARD_IA, and ONEZONE_IA. The following classes are not supported: All S3 Glacier classes (GLACIER and DEEP_ARCHIVE).</td>
</tr>
<tr>
<td>Versioning</td>
<td>AppConfig requires that the S3 object use versioning.</td>
</tr>
</tbody>
</table>

Configuring permissions for a configuration stored as an Amazon S3 object

When you create a configuration profile for a configuration stored as an S3 object, you must specify an ARN for an IAM role that gives AppConfig permission to get the object. The role must include the following permissions.

Permissions to access the S3 object
- s3:GetObject
- s3:GetObjectVersion

Permissions to list S3 buckets
s3:ListAllMyBuckets

Permissions to access the S3 bucket where the object is stored
- s3:GetBucketLocation
Complete the following procedure to create a role that enables AppConfig to get a configuration stored in an S3 object.

Creating the IAM Policy for Accessing an S3 Object

Use the following procedure to create an IAM policy that enables AppConfig to get a configuration stored in an S3 object.

To create an IAM policy for accessing an S3 object

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy.
3. On the Create policy page, choose the JSON tab.
4. Update the following sample policy with information about your S3 bucket and configuration object. Then paste the policy into the text field on the JSON tab.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::my-bucket/my-configurations/my-configuration.json"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetBucketVersioning",
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": ["arn:aws:s3:::my-bucket"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```
5. Choose Review policy.
6. On the Review policy page, type a name in the Name box, and then type a description.
7. Choose Create policy. The system returns you to the Roles page.

Creating the IAM Role for Accessing an S3 Object

Use the following procedure to create an IAM role that enables AppConfig to get a configuration stored in an S3 object.
To create an IAM role for accessing an Amazon S3 object

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
3. On the Select type of trusted entity section, choose AWS service.

4. In the Choose a use case section, under Common use cases, choose EC2, and then choose Next: Permissions.
5. On the Attach permissions policy page, in the search box, enter the name of the policy you created in the previous procedure.
6. Choose the policy and then choose Next: Tags.
7. On the Add tags (optional) page, enter a key and an optional value, and then choose Next:Review.
8. On the Review page, type a name in the Role name field, and then type a description.
9. Choose Create role. The system returns you to the Roles page.
10. On the Roles page, choose the role you just created to open the Summary page. Note the Role Name and Role ARN. You will specify the role ARN when you create the configuration profile later in this topic.
Creating a Trust Relationship

Use the following procedure to configure the role you just created to trust AppConfig.

To add a trust relationship

1. In the Summary page for the role you just created, choose the Trust Relationships tab, and then choose Edit Trust Relationship.
2. Delete "ec2.amazonaws.com" and add "appconfig.amazonaws.com", as shown in the following example.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "appconfig.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```
3. Choose Update Trust Policy.

About configuration store quotas and limitations

You can store configurations in one of the following locations:

- Objects in an Amazon Simple Storage Service (Amazon S3) bucket
- Parameters in Parameter Store
- Documents in the Systems Manager document store

These configuration stores have the following quotas and limitations.

<table>
<thead>
<tr>
<th></th>
<th>S3</th>
<th>Parameter Store</th>
<th>Document store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration size</td>
<td>1 MB</td>
<td>4 KB (free tier) / 8 KB (advanced</td>
<td>64 KB</td>
</tr>
<tr>
<td>limit</td>
<td>Enforced by AppConfig, not S3.</td>
<td>parameters)</td>
<td></td>
</tr>
<tr>
<td>Resource storage</td>
<td>Unlimited</td>
<td>10,000 parameters (free tier) / 100,000</td>
<td>500 documents</td>
</tr>
<tr>
<td>limit</td>
<td></td>
<td>parameters (advanced parameters)</td>
<td></td>
</tr>
<tr>
<td>Server-side encryption</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AWS CloudFormation</td>
<td>Not for creating or</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>support</td>
<td>updating data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validate create or</td>
<td>Not supported</td>
<td>Regexp supported</td>
<td>JSON Schema required for all put and update API actions</td>
</tr>
<tr>
<td>update API actions</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
About validators

When you create a configuration and configuration profile using the procedure in this topic, you can specify up to two validators. A validator ensures that your configuration data is syntactically and semantically correct. You can create validators in either JSON Schema or as an AWS Lambda function.

Important

Configuration data stored in SSM documents must validate against an associated JSON Schema before you can add the configuration to the system. SSM parameters do not require a validation method, but we recommend that you create a validation check for new or updated SSM parameter configurations by using AWS Lambda.

JSON Schema Validators

If you create a configuration in an SSM document by using the procedure in this topic, then you must specify or create a JSON Schema for that configuration. A JSON Schema defines the allowable properties for each application configuration setting. The JSON Schema functions like a set of rules to ensure that new or updated configuration settings conform to the best practices required by your application. Here is an example.

```json
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "title": "$id$",
   "description": "BasicFeatureToggle-1",
   "type": "object",
   "additionalProperties": false,
   "patternProperties": {
      "[^\s\$]+": {
         "type": "boolean"
      }
   },
   "minProperties": 1
}
```

When you create the configuration by using the procedure in this topic, AppConfig verifies that the configuration conforms to the schema requirements. If it doesn't, Systems Manager returns a validation error.

Note

AppConfig supports JSON Schema version 4.X for inline schema. If your application configuration requires a different version of JSON Schema, then you must create a Lambda validator.

AWS Lambda Validators

Lambda function validators must be configured with the following event schema. AppConfig uses this schema to invoke the Lambda function. The content is a base64-encoded string, and the URI is a string.

```json
{
   "content": Base64EncodedByteString,
   "uri": "The uri of the configuration"
}
```
AppConfig verifies that the Lambda X-Amz-Function-Error header is set in the response. Lambda sets this header if the function throws an exception. For more information about X-Amz-Function-Error, see Error Handling and Automatic Retries in AWS Lambda in the AWS Lambda Developer Guide.

Here is a simple example of a Lambda response code for a successful validation.

```python
import json
def handler(event, context):
    #Add your validation logic here
    print("We passed!")
```

Here is a simple example of a Lambda response code for an unsuccessful validation.

```python
def handler(event, context):
    #Add your validation logic here
    raise Exception("Failure!")
```

AppConfig calls your validation Lambda when calling the StartDeployment and ValidateConfigurationActivity API actions. You must provide appconfig.amazonaws.com permissions to invoke your Lambda. For more information, see Granting Function Access to AWS Services.

### About the configuration profile IAM role

You can create the IAM role that provides access to the configuration data by using AppConfig, as described in the following procedure. Or you can create the IAM role yourself and choose it from a list.

If you create the role by using AppConfig, the system creates the role and specifies one of the following permissions policies, depending on which type of configuration source you choose.

#### Configuration source is an SSM document

```json
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "ssm:GetDocument"
       ],
       "Resource": [
       ]
     }
   ]
}
```

#### Configuration source is a Parameter Store parameter

```json
{
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Action": [
         "ssm:GetParameter"
       ],
       "Resource": [
         "Arn:aws:ssm:AWS-Region:account-number:parameter/parameter-name"
       ]
     }
   ]
}
```
If you create the role by using AppConfig, the system also creates the following trust relationship for the role.

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

Creating a configuration and a configuration profile

Use the following procedure to create an AppConfig configuration and a configuration profile by using the AWS Systems Manager console.

To create a configuration profile

2. On the Applications tab, choose the application you created in Create an AppConfig configuration (p. 183) and then choose the Configuration profiles tab.
3. Choose Create configuration profile.
4. For Name, enter a name for the configuration profile.
5. For Description, enter information about the configuration profile.
6. In the Service role section, choose New service role to have AppConfig create the IAM role that provides access to the configuration data. AppConfig automatically populates the Role name field based on the name you entered earlier. Or, to choose a role that already exists in IAM, choose Existing service role. Choose the role by using the Role ARN list.
7. In the Tags section, enter a key and an optional value. You can specify a maximum of 50 tags for a resource.
8. On the Select configuration source page, choose an option.
9. If you selected Amazon S3 object, then enter the object URI. If you selected AWS Systems Manager parameter, then choose the name of the parameter from the list. If you selected AWS Systems Manager document, then complete the following steps.

a. In the Document source section, choose either Saved document or New document.

b. If you choose Saved document, then choose the SSM document from the list. If you choose New document, the Details and Content sections appear.

c. In the Details section, enter a name for the new application configuration.

d. For the Application configuration schema section, either choose the JSON schema using the list or choose Create schema. If you choose Create schema, Systems Manager opens the Create schema page. Enter the schema details in the Content section, and then choose Create schema.

e. For Application configuration schema version either choose the version from the list or choose Update schema to edit the schema and create a new version.
f. In the **Content** section, choose either **YAML** or **JSON** and then enter the configuration data in the field.

![Content section with YAML and JSON options]

- **YAML**: Specify document content in YAML format.
- **JSON**: Specify document content in JSON format.

```json
"AccessList": [
  {
    "user_name": "Mateo_Jackson"
  },
  {
    "user_name": "Jane_Doe"
  }
]
```

g. Choose **Next**.

10. On the **Add validators** page, choose either **JSON Schema** or **AWS Lambda**. If you choose **JSON Schema**, enter the JSON Schema in the field. If you choose **AWS Lambda**, choose the function Amazon Resource Name (ARN) and the version from the list.

![Add validators page]

**Important**
Configuration data stored in SSM documents must validate against an associated JSON Schema before you can add the configuration to the system. SSM parameters do not require a validation method, but we recommend that you create a validation check for new or updated SSM parameter configurations by using AWS Lambda.
11. Choose **Create configuration profile**.

AppConfig creates the configuration profile. Proceed to **Step 4: Creating a deployment strategy (p. 194)**.

**Step 4: Creating a deployment strategy**

An AWS AppConfig deployment strategy defines the following important aspects of a configuration deployment.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deployment type</strong></td>
<td>Deployment type defines how the configuration deploys or rolls out. AppConfig supports <strong>Linear</strong> and <strong>Exponential</strong> deployment types.</td>
</tr>
<tr>
<td></td>
<td><strong>Linear</strong>: For this type, AppConfig processes the deployment by increments of the growth factor evenly distributed over the deployment time. For example, a linear deployment that uses a step percentage of 20 initially makes the configuration available to 20 percent of the targets. After 1/5th of the deployment time has passed, the system updates the percentage to 40 percent. This continues until 100% of the targets are set to receive the deployed configuration.</td>
</tr>
<tr>
<td></td>
<td><strong>Exponential</strong>: For this type, AppConfig processes the deployment exponentially using the following formula: $G \times (2^N)$. In this formula, $G$ is the step percentage specified by the user and $N$ is the number of steps until the configuration is deployed to all targets. For example, if you specify a growth factor of 2, then the system rolls out the configuration as follows:</td>
</tr>
</tbody>
</table>
|                              | $2 \times (2^0)$  
|                              | $2 \times (2^1)$  
|                              | $2 \times (2^2)$  
|                              | Expressed numerically, the deployment rolls out as follows: 2% of the targets, 4% of the targets, 8% of the targets, and continues until the configuration has been deployed to all targets. |
| **Step percentage (growth factor)** | This setting specifies the percentage of callers to target during each step of the deployment.  
**Note**  
In the SDK and the AppConfig API Reference, step percentage is called growth factor. |
| **Deployment time**          | This setting specifies an amount of time during which AppConfig deploys to hosts. This is not a
Working with AWS AppConfig

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>timeout value. It is a window of time during which the deployment is processed in intervals.</td>
</tr>
<tr>
<td>Bake time</td>
<td>This setting specifies the amount of time AppConfig monitors for Amazon CloudWatch alarms before proceeding to the next step of a deployment or before considering the deployment to be complete. If an alarm is triggered during this time, AppConfig rolls back the deployment. You must configure permissions for AppConfig to roll back based on CloudWatch alarms. For more information, see (Optional) Configuring permissions for rollback based on CloudWatch alarms (p. 180).</td>
</tr>
</tbody>
</table>

**Predefined deployment strategies**

AppConfig includes predefined deployment strategies to help you quickly deploy a configuration. Instead of creating your own strategies, you can choose one of the following when you deploy a configuration.

<table>
<thead>
<tr>
<th>Deployment strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppConfig.AllAtOnce</td>
<td><strong>Quick:</strong></td>
</tr>
<tr>
<td></td>
<td>This strategy deploys the configuration to all targets immediately. The system monitors for Amazon CloudWatch alarms for 10 minutes. If no alarms are received in this time, the deployment is complete. If an alarm is triggered during this time, AppConfig rolls back the deployment.</td>
</tr>
<tr>
<td>AppConfig.Linear50PercentEvery30Seconds</td>
<td><strong>Testing/Demonstration:</strong></td>
</tr>
<tr>
<td></td>
<td>This strategy deploys the configuration to half of all targets every 30 seconds for a one-minute deployment. The system monitors for Amazon CloudWatch alarms for 1 minute. If no alarms are received in this time, the deployment is complete. If an alarm is triggered during this time, AppConfig rolls back the deployment.</td>
</tr>
<tr>
<td></td>
<td>We recommend using this strategy only for testing or demonstration purposes because it has a short duration and bake time.</td>
</tr>
<tr>
<td>AppConfig.Canary10Percent20Minutes</td>
<td><strong>AWS Recommended:</strong></td>
</tr>
<tr>
<td></td>
<td>This strategy processes the deployment exponentially using a 10% growth factor over 20 minutes. The system monitors for Amazon CloudWatch alarms for 10 minutes. If no alarms are received in this time, the deployment is complete. If an alarm is triggered during this time, AppConfig rolls back the deployment.</td>
</tr>
</tbody>
</table>

195
Deployment strategy | Description
---------------------|--------------------------------------------------
|                     | We recommend using this strategy for production deployments because it aligns with AWS best practices for configuration deployments.

Create a deployment strategy

You can create a maximum of 20 deployment strategies. When you deploy a configuration, you can choose the deployment strategy that works best for the application and the environment.

Use the following procedure to create an AppConfig deployment strategy by using the AWS Systems Manager console.

To create a deployment strategy

2. In the navigation pane, choose AppConfig.
3. Choose the Deployment Strategies tab, and then choose Create deployment strategy.
4. For Name, enter a name for the deployment strategy.
5. For Description, enter information about the deployment strategy.
6. For Deployment type, choose a type.
7. For Step percentage, choose the percentage of callers to target during each step of the deployment.
8. For Deployment time, enter the total duration for the deployment in minutes or hours.
9. For Bake time, enter the total time, in minutes or hours, to monitor for Amazon CloudWatch alarms before proceeding to the next step of a deployment or before considering the deployment to be complete.
10. In the Tags section, enter a key and an optional value. You can specify a maximum of 50 tags for a resource.
11. Choose Create deployment strategy.

AppConfig creates the deployment strategy. Proceed to Step 5: Deploying a configuration (p. 196).

Step 5: Deploying a configuration

Starting a deployment in AWS AppConfig calls the StartDeployment API action. This call includes the IDs of the AppConfig application, the environment, the configuration profile, and (optionally) the configuration data version to deploy. The call also includes the ID of the deployment strategy to use, which determines how the configuration data is deployed.

AppConfig monitors the distribution to all hosts and reports status. If a distribution fails, then AppConfig rolls back the configuration.

Deploy a configuration

Use the following procedure to deploy an AppConfig configuration by using the AWS Systems Manager console.

To deploy a configuration by using the console

2. In the navigation pane, choose AppConfig.
3. On the Applications tab, choose an application, and then choose View details.
4. On the **Environments** tab, choose an environment, and then choose **View details**.
5. Choose **Start deployment**.
6. For **Configuration**, choose a configuration from the list.
7. Depending on the source of your configuration, use the **Document version** or **Parameter version** list to choose the version you want to deploy.
8. For **Deployment strategy**, choose a strategy from the list.
9. For **Deployment description**, enter a description.
10. In the **Tags** section, enter a key and an optional value. You can specify a maximum of 50 tags for a resource.
11. Choose **Start deployment**.

**Step 6: Receiving the configuration**

You must configure a client to receive configuration updates by integrating with the **GetConfiguration** API action. You can integrate using the AWS SDK. The following AWS CLI command demonstrates how to receive a configuration. This call includes the IDs of the AppConfig application, the environment, the configuration profile, and a unique client ID. The configuration content is saved to the output filename.

**Note**
The **client-id** parameter in the following command is a unique, user-specified ID to identify the client for the configuration. This ID enables AppConfig to deploy the configuration in intervals, as defined in the deployment strategy.

```
aws appconfig get-configuration \
    --application application_name_or_ID \
    --environment environment_name_or_ID \
    --configuration configuration_profile_name_or_ID \
    --client-id client_ID \
    output_filename
```

The system responds with information in the following format.

```
{
    "ConfigurationVersion":"configuration version",
    "ContentType":"content type"
}
```

**Important**
AWS AppConfig uses the value of the **ClientConfigurationVersion** parameter to identify the configuration version on your clients. If you don’t send **ClientConfigurationVersion** with each call to **GetConfiguration**, your clients receive the current configuration. You are charged each time your clients receive a configuration.

To avoid excess charges, we recommend that you include the **ClientConfigurationVersion** value with every call to **GetConfiguration**. This value must be saved on your client. Subsequent calls to **GetConfiguration** must pass this value by using the **ClientConfigurationVersion** parameter, as shown here.

Sending **ConfigurationVersion** during subsequent polling for configuration updates is similar to the concept of **HTTP ETags**.

```
aws appconfig get-configuration \
    --application application_name_or_ID \
    --environment environment_name_or_ID \
    --configuration configuration_profile_name_or_ID \
    --client-configuration-version previous_configuration_version_value \
    --client-id client_ID \
```
**Note**
We recommend tuning the polling frequency of your GetConfiguration API calls based on your budget, the expected frequency of your configuration deployments, and the number of targets for a configuration.

**AWS Systems Manager Parameter Store**

AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, Amazon Machine Image (AMI) IDs, and license codes as parameter values. You can store values as plain text or encrypted data. You can reference Systems Manager parameters in your scripts, commands, SSM documents, and configuration and automation workflows by using the unique name that you specified when you created the parameter.

**Important**
Do not store sensitive data in a String or StringList parameter. For all sensitive data that must remain encrypted, use only the SecureString parameter type. For more information, see Parameter types and examples (p. 199) and SecureString parameters (p. 206).

Parameter Store offers the following benefits and features.

- Use a secure, scalable, hosted secrets management service with no servers to manage.
- Improve your security posture by separating your data from your code.
- Store configuration data and encrypted strings in hierarchies and track versions.
- Control and audit access at granular levels.
- Configure change notifications and trigger automated actions for both parameters and parameter policies.
- Tag parameters individually, and then secure access from different levels, including operational, parameter, Amazon EC2 tag, and path levels.
- Validation of ID format when you specify an Amazon Machine Image (AMI) ID as a parameter value.
- Reference AWS Secrets Manager secrets by using Parameter Store parameters.
- Use Parameter Store parameters with other Systems Manager capabilities and AWS services to retrieve secrets and configuration data from a central store.

Parameters work with Systems Manager capabilities such as Run Command, State Manager, and Automation. You can also reference parameters in a number of other AWS services, including the following:

- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- AWS Secrets Manager
- AWS Lambda
- AWS CloudFormation
- AWS CodeBuild
- AWS CodePipeline
- AWS CodeDeploy
- Configure integration with the following AWS services for encryption, notification, monitoring, and auditing:
  - AWS Key Management Service (AWS KMS)
  - Amazon Simple Notification Service (Amazon SNS)
• Amazon CloudWatch
• AWS CloudTrail

Tagging a parameter

You can tag your parameters to help you quickly identify one or more parameters based on the tags you've assigned to them. For example, you can tag parameters for specific environments, departments, users, groups, or periods. You can also restrict access to parameters by creating an IAM policy that specifies the tags that a user or group can access. For more information, see Tagging Systems Manager parameters (p. 1180).

Topics
• Parameter types and examples (p. 199)
• SecureString parameters (p. 206)
• Native parameter support for Amazon Machine Image IDs (p. 210)
• Public parameters (p. 212)
• Getting started with Parameter Store (p. 224)
• Working with parameters (p. 240)
• Parameter Store walkthroughs (p. 266)

Parameter types and examples

A Parameter Store parameter is any piece of data that is saved in Parameter Store, such as a block of text, a list of names, a password, an Amazon Machine Image (AMI) ID, a license key, and so on. You can centrally and securely reference this data in your scripts, commands, and SSM documents.

Important
Do not store sensitive data in a String or StringList parameter. For all sensitive data that must remain encrypted, use only the SecureString parameter type.
For more information, see SecureString parameters (p. 206).

When you reference a parameter, you specify the parameter name by using the following convention.

```
{{ssm:parameter-name}}
```

Topics
• Parameter types (p. 199)
• Parameter examples (AWS CLI) (p. 201)
• Integration examples from the community (p. 204)

Parameter types

Parameter Store provides support for three types of parameters: String, StringList, and SecureString.

With one exception, when you create or update a parameter, you enter the parameter value as plain text, and Parameter Store performs no validation on the text you enter. For String parameters, however, you can specify the data type as aws:ec2:image, and Parameter Store validates that the value you enter is the proper format for an Amazon EC2 AMI; for example: ami-12345abcdefEXAMPLE.

String

By default, String parameters consist of any block of text you enter. For example:
You can also use the `DataType` option to validate that the parameter value you enter is a properly formatted Amazon EC2 AMI ID, as shown in the following example AWS CLI command.

**Linux**

Parameter not in a hierarchy:

```bash
aws ssm put-parameter
  --name "golden-ami"
  --type "String"
  --data-type "aws:ec2:image"
  --value "ami-12345abcdeEXAMPLE"
```

Parameter in a hierarchy:

```bash
aws ssm put-parameter
  --name \\
  "\amis\linux\golden-ami"
  --type "String"
  --data-type "aws:ec2:image"
  --value "ami-12345abcdeEXAMPLE"
```

**Windows**

Parameter not in a hierarchy:

```bash
aws ssm put-parameter ^
  --name "golden-ami" ^
  --type "String" ^
  --data-type "aws:ec2:image" ^
  --value "ami-12345abcdeEXAMPLE"
```

Parameter in a hierarchy:

```bash
aws ssm put-parameter ^
  --name \\
  "\amis\windows\golden-ami" ^
  --type "String" ^
  --data-type "aws:ec2:image" ^
  --value "ami-12345abcdeEXAMPLE"
```

You do not need to specify a data type in any other cases.

**StringList**

StringList parameters contain a comma-separated list of values, as shown in the following examples.

Monday, Wednesday, Friday

CSV, TSV, CLF, ELF, JSON

**SecureString**

The SecureString parameter type can be used for textual data that you want to encrypt, such as passwords, application secrets, confidential configuration data, or any other types of data you
need to protect. SecureString data is encrypted and decrypted using a AWS Key Management Service (KMS) key. You can use either a default KMS key provided by AWS or create and use your own customer master key (CMK). (Use your own CMK if you need to restrict user access to SecureString parameters. For information, see IAM permissions for using AWS default keys and customer managed keys (p. 207).)

There is no charge from Parameter Store to create a SecureString parameter, but charges for use of AWS Key Management Service encryption do apply. For information, see AWS Key Management Service pricing.

**Note**
Parameter Store is also integrated with AWS Secrets Manager. You can retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters. For more information, see Referencing AWS Secrets Manager secrets from Parameter Store parameters (p. 98) in this guide.

For more information about SecureString parameters, see SecureString parameters (p. 206).

### Parameter examples (AWS CLI)

#### Creating parameters

The following example creates a plain-text String parameter.

**Linux**

```
aws ssm put-parameter \\
  --name "MyStringTextParameter" \\
  --type "String" \\
  --value "Text parameter test"
```

**Windows**

```
aws ssm put-parameter ^
  --name "MyStringTextParameter" ^
  --type "String" ^
  --value "Text parameter test"
```

The following example creates a String parameter that specifies the data type as `aws:ec2:image`.

**Linux**

```
aws ssm put-parameter \\
  --name "MyStringAMIParameter" \\
  --type "String" \\
  --data-type "aws:ec2:image" \\
  --value "ami-12345abcdeEXAMPLE"
```

**Windows**

```
aws ssm put-parameter ^
  --name "MyStringAMIParameter" ^
  --type "String" ^
  --data-type "aws:ec2:image" ^
  --value "ami-12345abcdeEXAMPLE"
```
For information about using the data type `aws:ec2:image`, see Parameter types (p. 199) and Native parameter support for Amazon Machine Image IDs (p. 210).

The following example creates a `StringList` parameter:

**Linux**

```
aws ssm put-parameter \
 --name "MyStringListParameter" \
 --type "StringList" \
 --value "North,South,East,West"
```

**Windows**

```
aws ssm put-parameter ^
 --name "MyStringListParameter" ^
 --type "StringList" ^
 --value "North,South,East,West"
```

For more examples of creating and updating parameters using the AWS CLI, see Create a Systems Manager parameter (AWS CLI) (p. 251) and put-parameter in the AWS Systems Manager section of the AWS CLI Command Reference.

**Parameters in Run Command commands**

The following example command includes a Systems Manager parameter named `DNS-IP`. The value of this parameter is simply the IP address of an instance. This example uses an AWS CLI command to echo the parameter value.

**Linux**

```
aws ssm send-command \
 --document-name "AWS-RunPowerShellScript" \
 --document-version "1" \
 --targets "Key=instanceids,Values=i-02573cafcfEXAMPLE" \
 --parameters "commands='echo {{ssm:DNS-IP}}'" \
 --timeout-seconds 600 \
 --max-concurrency "50" \
 --max-errors "0" \
 --region us-east-2
```

**Windows**

```
aws ssm send-command ^
 --document-name "AWS-RunPowerShellScript" ^
 --document-version "1" ^
 --targets "Key=instanceids,Values=i-02573cafcfEXAMPLE" ^
 --parameters "commands='echo {{ssm:DNS-IP}}'" ^
 --timeout-seconds 600 ^
 --max-concurrency "50" ^
 --max-errors "0" ^
 --region us-east-2
```

The next example command uses a SecureString parameter named `SecurePassword`. The command `commands=["$secure = (Get-SSMParameterValue -Names SecurePassword -WithDecryption $true).Parameters[0].Value","net user administrator $secure"]` retrieves and decrypts the value of the SecureString parameter, and then resets the local administrator password without having to pass the password in clear text.
AWS Systems Manager User Guide
Parameter types and examples

### Linux

```bash
aws ssm send-command 
  --document-name "AWS-RunPowerShellScript" 
  --document-version "1" 
  --targets "Key=instanceids,Values=i-02573cafcfEXAMPLE" 
  --parameters "commands=["$secure = (Get-SSMParameterValue -Names SecurePassword -WithDecryption $True).Parameters[0].Value','net user administrator $secure"]" 
  --timeout-seconds 600 
  --max-concurrency "50" 
  --max-errors "0" 
  --region us-east-2
```

### Windows

```bash
aws ssm send-command ^
  --document-name "AWS-RunPowerShellScript" ^
  --document-version "1" ^
  --targets "Key=instanceids,Values=i-02573cafcfEXAMPLE" ^
  --parameters "commands=["$secure = (Get-SSMParameterValue -Names SecurePassword -WithDecryption $True).Parameters[0].Value','net user administrator $secure"]" ^
  --timeout-seconds 600 ^
  --max-concurrency "50" ^
  --max-errors "0" ^
  --region us-east-2
```

You can also reference Systems Manager parameters in the **Parameters** section of an SSM document, as shown in the following example.

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

Don't confuse the similar syntax for **local parameters** used in the **runtimeConfig** section of SSM documents with Parameter Store parameters. A local parameter isn't the same as a Systems Manager parameter. You can distinguish local parameters from Systems Manager parameters by the absence of the **ssm:** prefix:

```json
"runtimeConfig":{
  "aws:runShellScript":{
    "properties":{
      "id":"0.aws:runShellScript",
      "runCommand":"{{ commands }}",
```
"workingDirectory": "{{ workingDirectory }}",
"timeoutSeconds": "{{ executionTimeout }}"

**Note**
SSM documents currently don’t support references to SecureString parameters. This means that to use SecureString parameters with, for example, Run Command, you have to retrieve the parameter value before passing it to Run Command, as shown in the following examples:

**Linux**

```bash
value=$(aws ssm get-parameters --names parameter-name --with-decryption)
aws ssm send-command \
  --name AWS-JoinDomain \
  --parameters password=$value \
  --instance-id instance-id
```

**Windows**

```bash
aws ssm send-command ^
  --name AWS-JoinDomain ^
  --parameters password=$value ^
  --instance-id instance-id
```

**Tools for Windows PowerShell**

```powershell
$secure = (Get-SSMParameterValue -Names parameter-name -WithDecryption $True).Parameters[0].Value | ConvertTo-SecureString -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential -argumentlist userName,$secure
```

**Integration examples from the community**

The following section provide links to blog posts, articles, and community-provided examples for using Parameter Store parameters.

**Note**
These links are provided for informational purposes only, and should not be considered either a comprehensive list or an endorsement of the content of the examples. AWS is not responsible for the content or accuracy of external content.

- **A complete guide to using the AWS Systems Manager Parameter Store**
  
  Sean Ziegler offers a concise overview of Parameter Store functionality and provides a Boto3 example for interacting with Parameter Store.
  
  *May 23, 2020*

- **Keep Your Secrets Safe with AWS Systems Manager Parameter Store and Node**
  
  Amir Boroumand walks us through how to save and retrieve a password using the Parameter Store with Node.
  
  *October 19, 2019*
• Using SSM Parameters with CloudFormation Templates and Terraform Projects

J Cole Morrison demonstrates how to create parameters for use in AWS CloudFormation templates and Terraform projects so you can reference values with less human error.

August 29, 2019

• Using AWS Systems Manager Parameter Store for Configuration

Learn how to store and retrieve application configuration settings at runtime for an application instead of hard-coding the configuration values into a sample application's code and configuration files. The sample application, a simple .NET Core console application, shows how to use the AWS SDK for .NET to retrieve configuration values from Parameter Store.

March 8, 2019

• Using AWS Systems Manager Parameter Store SecureString parameters in AWS CloudFormation templates

Learn how to use plain text and SecureString parameters in your AWS CloudFormation templates through the use of dynamic references to fetch parameter values.

October 5, 2018

• Use parameter labels for easy configuration update across environments

Learn how to use a parameter label as an alias for a parameter version. You can group parameter versions across hierarchies using labels, and you can deploy new updates to parameters across environments using hierarchies and labels.

July 26, 2018

• Integrating AWS CloudFormation with AWS Systems Manager Parameter Store

Learn how to use Parameter Store parameters in your AWS CloudFormation templates to simplify stack updates involving parameters and achieve consistency by using values stored in Parameter Store. With this integration, your code remains untouched while the stack update operation automatically picks up the latest parameter value.

December 28, 2017

• The Right Way to Store Secrets using Parameter Store

Evan Johnson presents a case study in how Segment centrally and securely manages their secrets using a combination of Parameter Store, lots of Terraform code, and chamber, with a focus on getting up and running with Parameter Store in production.

August 27, 2017

• Join a Microsoft Active Directory Domain with Parameter Store and Amazon EC2 Systems Manager Documents

Read about a scenario for centralized configuration management, with an example of joining EC2 instances to a Microsoft Active Directory. This post shows you how to launch an EC2 instance that consumes and uses configuration values stored as Parameter Store parameters to join your Active Directory domain.

June 19, 2017

• Use Parameter Store to Securely Access Secrets and Config Data in AWS CodeDeploy

Learn how to simplify your CodeDeploy workflows by using Parameter Store to store and reference a configuration secret. Doing so not only improves your security posture, but also automates your deployment because you don’t have to manually change configuration data in your source code.
SecureString parameters

A SecureString parameter is any sensitive data that needs to be stored and referenced in a secure manner. If you have data that you don't want users to alter or reference in plain text, such as passwords or license keys, create those parameters using the SecureString datatype.

Important
Do not store sensitive data in a String or StringList parameter. For all sensitive data that must remain encrypted, use only the SecureString parameter type. For more information, see Parameter types and examples (p. 199).

We recommend using SecureString parameters for the following scenarios.

• You want to use data/parameters across AWS services without exposing the values as plain text in commands, functions, agent logs, or AWS CloudTrail logs.
• You want to control who has access to sensitive data.
• You want to be able to audit when sensitive data is accessed (AWS CloudTrail).
• You want to encrypt your sensitive data and you want to bring your own encryption keys to manage access.

Important
Only the value of a SecureString parameter is encrypted. Parameter names, descriptions, and other properties are not encrypted.

AWS KMS encryption and pricing
If you choose the `SecureString` type when you create a parameter, then Parameter Store uses an AWS Key Management Service (KMS) customer master key (CMK) to encrypt the parameter value. KMS uses either a customer managed CMK or an AWS-managed CMK when encrypting the parameter value.

There is no charge from Parameter Store to create a `SecureString` parameter, but charges for use of AWS Key Management Service encryption do apply. For information, see AWS Key Management Service pricing.

For more information about AWS managed and customer managed CMKs, see AWS Key Management Service Concepts in the AWS Key Management Service Developer Guide. For more information about Parameter Store and AWS KMS encryption, see How AWS Systems Manager Parameter Store Uses AWS KMS.

**Note**
To view a CMK, use the AWS KMS DescribeKey operation. This AWS CLI example uses DescribeKey to view an AWS-managed CMK.

```bash
aws kms describe-key --key-id alias/aws/ssm
```

### IAM permissions for using AWS default keys and customer managed keys

Parameter Store `SecureString` parameters are encrypted and decrypted using AWS Key Management Service (AWS KMS) keys. You can choose to encrypt your `SecureString` parameters using either a customer master key (CMK) or the default KMS key provided by AWS.

When using a customer managed key, the IAM policy that grants a user access to a parameter or parameter path must provide explicit `kms:Encrypt` permissions for the key. For example, the following policy allows a user to create, update, and view `SecureString` parameters that begin with "prod-" in the specified Region and account.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ssm:PutParameter",
            "ssm:GetParameter",
            "ssm:GetParameters"
         ],
         "Resource": [
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "kms:Decrypt",
            "kms:Encrypt",
            "kms:GenerateDataKey"
         ],
         "Resource": [
            "arn:aws:kms:us-east-2:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE"
         ]
      }
   ]
}
```
The `kms:GenerateDataKey` permission is required for creating encrypted advanced parameters using the specified customer managed key.

By contrast, all users within the customer account have access to the default AWS managed key. If you use this default key to encrypt `SecureString` parameters and do not want users to work with `SecureString` parameters, their IAM policies must explicitly deny access to the default key, as demonstrated in the following policy example.

**Note**
You can locate the ARN of the default key in the AWS KMS console on the AWS managed keys page. The default key is the one identified with `aws/ssm` in the **Alias** column.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "default-key-ARN"
      ]
    }
  ]
}
```

If you require fine-grained access control over the `SecureString` parameters in your account, you should use a customer managed CMK to protect and restrict access to these parameters. We also recommend using AWS CloudTrail to monitor `SecureString` parameter activities.

For more information, see the following topics.

- Policy Evaluation Logic in the *IAM User Guide*
- Using key policies in AWS KMS in the *AWS Key Management Service Developer Guide*
- Viewing Events with CloudTrail Event History in the *AWS CloudTrail User Guide*

**Create a SecureString parameter using the default AWS managed key**

If you create a `SecureString` parameter by using the AWS-managed AWS KMS key in your account and Region, then you don’t have to provide a value for the `--key-id` parameter.

The following AWS CLI example shows the command to create a new `SecureString` parameter in Parameter Store without the `--key-id` parameter:

**Linux**

```
aws ssm put-parameter \ 
  --name parameter-name \ 
  --value "parameter-value" \ 
  --type SecureString
```

**Windows**

```
aws ssm put-parameter ^
```
Create a SecureString parameter using a customer managed CMK

To use a customer managed CMK instead of the AWS-managed CMK assigned to your account, you must specify the key by using the --key-id parameter. The parameter supports the following KMS parameter formats.

- Key ARN example:
  
  `arn:aws:kms:us-east-2:123456789012:key/12345678-1234-1234-1234-123456789012`

- Alias ARN example:
  
  `arn:aws:kms:us-east-2:123456789012:alias/MyAliasName`

- Key ID example:
  
  `12345678-1234-1234-1234-123456789012`

- Alias Name example:
  
  `alias/MyAliasName`

You can create a customer managed CMK by using the AWS Management Console or the AWS KMS API. The following AWS CLI commands create a customer managed key in the current Region of your AWS account.

```
aws kms create-key
```

Use a command in the following format to create a SecureString parameter using the key you just created.

**Linux**

```
aws ssm put-parameter \
--name parameter-name \ 
--value "parameter-value" \ 
--type SecureString \ 
--key-id arn:aws:kms:us-east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

**Windows**

```
aws ssm put-parameter ^ 
--name parameter-name ^ 
--value "parameter-value" ^ 
--type SecureString ^ 
--key-id arn:aws:kms:us-east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

**Note**

You can manually create a parameter with an encrypted value. In this case, because the value is already encrypted, you don’t have to choose the SecureString parameter type. If you do choose SecureString, your parameter will be doubly encrypted.
AWS Systems Manager User Guide
Native parameter support for Amazon Machine Image IDs

By default, all SecureString values are displayed as cipher-text. To decrypt a SecureString value, a user must have permission to call the KMS Decrypt API action. For information about configuring KMS access control, see Authentication and Access Control for AWS KMS in the AWS Key Management Service Developer Guide.

Using SecureString parameters with other AWS services

You can also use SecureString parameters with other AWS services. In the following example, the AWS Lambda function retrieves a SecureString parameter by using the GetParameters API.

```python
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
```

Related topics

For an example of how to create and use a SecureString parameter, see Walkthrough: Create a SecureString parameter and join an instance to a Domain (PowerShell) (p. 272). For more information about using Systems Manager parameters with other AWS services, see the following blog posts.

- Managing Secrets for Amazon ECS Applications Using Parameter Store and IAM Roles for Tasks
- Use Parameter Store to Securely Access Secrets and Config Data in CodeDeploy
- Interesting Articles on Amazon EC2 Systems Manager Parameter Store

Native parameter support for Amazon Machine Image IDs

When you create a String parameter, you can specify the data type as `aws:ec2:image` to ensure that the parameter value you enter is a valid Amazon Machine Image (AMI) ID format.

Support for AMI ID formats lets you avoid updating all your scripts and templates with a new ID each time the AMI that you want to use in your processes changes. You can create a parameter with the data type `aws:ec2:image`, and for its value, enter the ID of an AMI. This is the AMI from which you currently want new instances to be created. You then reference this parameter in your templates, commands, and scripts.

For example, you can specify the parameter that contains your preferred AMI ID when you run the Amazon EC2 `run-instances` command.

Note

The user who runs this command must have IAM permissions that include the `ssm:GetParameters` API action in order for the parameter value to be validated. Otherwise, the parameter creation process fails.
You can also choose the Systems Manager containing your preferred AMI when you create an instance using the Amazon EC2 console. For more information, see Using an SSM parameter to find an AMI in the Amazon EC2 User Guide for Linux Instances.

When it's time to use a different AMI in your instance creation workflow, you need only update the parameter with the new AMI value, and Parameter Store again validates that you have entered an ID in the proper format.

**How AMI format validation works**

When you specify `aws:ec2:image` as the data type for a parameter, Systems Manager does not create the parameter immediately. It instead performs an asynchronous validation operation to ensure that the parameter value meets the formatting requirements for an AMI ID, and that the specified AMI is available in your AWS account.

It is important to note that a parameter version number may be generated before the validation operation is complete. That is, a parameter version number being generated alone isn't an indication that the operation has completed successfully.

To monitor whether your parameters are created successfully, we recommend using Amazon CloudWatch Events to send you notifications about your create and update parameter operations. These notifications report whether a parameter operation was successful or not. If an operation fails, the notification includes an error message that indicates the reason for the failure.

```json
{
    "version": "0",
    "id": "eed4a719-0fa4-6a49-80d8-8ac65EXAMPLE",
    "detail-type": "Parameter Store Change",
    "source": "aws.ssm",
    "account": "111122223333",
    "time": "2020-05-26T22:04:42Z",
    "region": "us-east-2",
    "resources": [
    ],
    "detail": {
        "exception": "Unable to Describe Resource",
        "dataType": "aws:ec2:image",
        "name": "golden-ami",
        "type": "String",
        "operation": "Create"
    }
}
```

For information about subscribing to Parameter Store events in CloudWatch Events, see Setting up notifications or trigger actions based on Parameter Store events (p. 226).

**Troubleshooting `aws:ec2:image` parameter creation**

Use the following information to help troubleshoot problems with creating `aws:ec2:image` data type parameters.

**Topics**
• CloudWatch Events reports the failure message "Unable to Describe Resource" (p. 212)
• New aws:ec2:image parameter isn't available (p. 212)

CloudWatch Events reports the failure message "Unable to Describe Resource"

**Problem:** You ran a command to create an aws:ec2:image parameter, but parameter creation failed. You receive a notification from CloudWatch Events that reports the exception "Unable to Describe Resource".

**Solution:** This message can indicate the following:

• You haven't been granted the IAM permission for the ec2:DescribeImages API action, or you lack permission to access the specific image referenced in the parameter. Contact an IAM user with administrator permissions in your organization to request the necessary permissions.
• The AMI ID you entered as a parameter value is not valid. Make sure you are entering the ID of an AMI that is available in the current AWS Region and account you are working in.

New aws:ec2:image parameter isn't available

**Problem:** You just ran a command to create an aws:ec2:image parameter and a version number was reported, but the parameter isn't available.

**Solution:** When you run the command to create a parameter that uses the aws:ec2:image data type, a version number is generated for the parameter right away, but the parameter format must be validated before the parameter is available. This process can take up to a few minutes. To monitor the parameter creation and validation process, you can do the following:

• Use Amazon CloudWatch Events to send you notifications about your create and update parameter operations. These notifications report whether a parameter operation was successful or not. For information about subscribing to Parameter Store events in CloudWatch Events, see Setting up notifications or trigger actions based on Parameter Store events (p. 226).
• In the Parameter Store section of the Systems Manager console, refresh the list of parameters periodically to check for the new or updated parameter details.
• Use the GetParameter command to check for the new or updated parameter. For example, using the AWS CLI:

```bash
aws ssm get-parameter --name MyParameter
```

For a new parameter, a ParameterNotFound message is returned until the parameter is validated. For an existing parameter that you are updating, information about the new version isn't included until the parameter is validated.

If you attempt to create or update the parameter again before the validation process completes, the system reports that validation is still in process. If the parameter isn't created or updated successfully, you can try again after five minutes have passed from the original attempt.

Public parameters

Some AWS services publish information about common artifacts as Systems Manager public parameters. For example, the Amazon Elastic Compute Cloud (Amazon EC2) service publishes information about Amazon Machines Images (AMIs) as public parameters.

You can call this information from your scripts and code by using the GetParametersByPath, GetParameter, and GetParameters API actions.
Related blog posts

- Query for AWS Regions, Endpoints, and More Using AWS Systems Manager Parameter Store
- Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store
- Query for the Latest Windows AMI Using AWS Systems Manager Parameter Store

The remainder of this topic describes how to call public parameters by using the AWS CLI.

Contents

- Calling AMI public parameters (p. 213)
- Calling the ECS optimized AMI public parameter (p. 215)
- Calling the EKS optimized AMI public parameter (p. 216)
- Calling public parameters for AWS services, Regions, endpoints, availability zones, and local zones (p. 216)

Calling AMI public parameters

Amazon EC2 AMI public parameters are available from the following paths:

- /aws/service/ami-amazon-linux-latest
- /aws/service/ami-windows-latest

You can view a list of all Linux AMIs in the current AWS Region by using the following command in the AWS CLI.

Linux

```bash
aws ssm get-parameters-by-path \
  --path /aws/service/ami-amazon-linux-latest \
  --query 'Parameters[].Name'
```

Windows

```bash
aws ssm get-parameters-by-path ^
  --path /aws/service/ami-amazon-linux-latest ^
  --query Parameters[].Name
```

The command returns information like the following.

```
["/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs",
 "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2",
 "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-s3",
 "/aws/service/ami-amazon-linux-latest/amzn-ami-minimal-hvm-x86_64-s3",
 "/aws/service/ami-amazon-linux-latest/amzn-ami-minimal-pv-x86_64-s3",
 "/aws/service/ami-amazon-linux-latest/amzn-ami-pv-x86_64-s3",
 "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2",
 "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-ebs",
 "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
 "/aws/service/ami-amazon-linux-latest/amzn2-ami-minimal-hvm-arm64-ebs",
 "/aws/service/ami-amazon-linux-latest/amzn2-ami-minimal-hvm-x86_64-ebs",
 "/aws/service/ami-amazon-linux-latest/amzn2-ami-minimal-pv-x86_64-ebs",
 "/aws/service/ami-amazon-linux-latest/amzn-ami-pv-x86_64-ebs",
]```
"/aws/service/ami-amazon-linux-latest/amzn2-ami-minimal-hvm-x86_64-ebs"
]

You can view details about these AMIs, including the AMI IDs and Amazon Resource Names (ARNs), by using the following command.

**Linux**

```bash
aws ssm get-parameters-by-path \
--path "/aws/service/ami-amazon-linux-latest" \
--region region
```

**Windows**

```bash
aws ssm get-parameters-by-path ^
--path "/aws/service/ami-amazon-linux-latest" ^
--region region
```

*region* represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported *region* values, see the Region column in Systems Manager service endpoints in the *Amazon Web Services General Reference*.

The command returns information like the following. This example output has been truncated for space.

```json
{
  "Parameters": [
    {
      "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs",
      "Type": "String",
      "Value": "ami-0d75cc1d706735521",
      "Version": 7,
      "LastModifiedDate": 1543873943.358,
      "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-ebs"
    },
    {
      "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2",
      "Type": "String",
      "Value": "ami-0cd3dfa4e37921605",
      "Version": 7,
      "LastModifiedDate": 1543873943.47,
      "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2"
    },
    {
      "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-s3",
      "Type": "String",
      "Value": "ami-0a0e3ff8af8d19497",
      "Version": 7,
      "LastModifiedDate": 1543873943.576,
      "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-s3"
    },
    {
      "Name": "/aws/service/ami-amazon-linux-latest/amzn-ami-minimal-hvm-x86_64-ebs",
      "Type": "String",
      "Value": "ami-0786a9626196d6dac",
      "Version": 7,
      "LastModifiedDate": 1543873943.682,
      "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/ami-amazon-linux-latest/amzn-ami-minimal-hvm-x86_64-ebs"
  ]
}
```
You can view details of a specific AMI by using the `GetParameters` API action with the full AMI name, including the path. Here is an example command.

**Linux**

```bash
aws ssm get-parameters \
  --names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 \
  --region us-west-2
```

**Windows**

```bash
aws ssm get-parameters ^
  --names /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2 ^
  --region us-west-2
```

The command returns the following information.

```json
{
  "Parameters": [
    {
      "Name": "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      "Type": "String",
      "Value": "ami-061392db613a6357b",
      "Version": 16,
      "LastModifiedDate": 1552519670.776,
      "ARN": "arn:aws:ssm:us-west-2::parameter/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2"
    }
  ],
  "InvalidParameters": []
}
```

**Calling the ECS optimized AMI public parameter**

The Amazon Elastic Container Service (Amazon ECS) service publishes the name of the latest Amazon ECS optimized AMI as a public parameter. Users are encouraged to use this AMI when creating a new Amazon EC2 cluster for Amazon ECS because the optimized AMI includes bug fixes and feature updates.

Use the following command to view the name of the latest Amazon ECS optimized AMI.

**Linux**

```bash
aws ssm get-parameters \
  --names /aws/service/ecs/optimized-ami/amazon-linux/recommended
```

**Windows**

```bash
aws ssm get-parameters ^
  --names /aws/service/ecs/optimized-ami/amazon-linux/recommended
```

The command returns information like the following.

```json
{
  "Parameters": [
    
  ],
  "InvalidParameters": []
}
```
Calling the EKS optimized AMI public parameter

The Amazon Elastic Kubernetes Service (Amazon EKS) service publishes the name of the latest Amazon EKS optimized AMI as a public parameter. Users are encouraged to use this AMI when adding nodes to an Amazon EKS cluster, as new releases include Kubernetes patches and security updates. Previously, to ensure you were using the latest AMI meant checking the Amazon EKS documentation and manually updating any deployment templates or resources with the new AMI ID.

Use the following command to view the name of the latest Amazon EKS optimized AMI.

**Linux**

```bash
aws ssm get-parameters \\n   --names /aws/service/eks/optimized-ami/1.14/amazon-linux-2/recommended
```

**Windows**

```bash
aws ssm get-parameters ^ \\n   --names /aws/service/eks/optimized-ami/1.14/amazon-linux-2/recommended
```

The command returns information like the following.

```
{
   "Parameters": [
   {
   "Name": "/aws/service/eks/optimized-ami/1.14/amazon-linux-2/recommended",
   "Type": "String",
   "Value": "{schema_version":2,"image_id":"ami-0b89776dcfa5f2dee",
   "image_name":"amazon-eks-node-1.14-v20200507","release_version":"1.14.9-20200507"},
   "Version": 11,
   "LastModifiedDate": "2020-05-11T17:34:51.990000-07:00",
   "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/eks/optimized-ami/1.14/amazon-linux-2/recommended"
   },
   "InvalidParameters": []
}
```

Calling public parameters for AWS services, Regions, endpoints, availability zones, and local zones

You can call AWS service, Region, endpoint, and availability zone public parameters by using the following path.
View active AWS Regions

You can view a list of all active AWS Regions by using the following command in the AWS CLI.

Linux

```bash
aws ssm get-parameters-by-path
   --path /aws/service/global-infrastructure/regions
   --query 'Parameters[].Name'
```

Windows

```bash
aws ssm get-parameters-by-path
   --path /aws/service/global-infrastructure/regions
   --query Parameters[].Name
```

The command returns information like the following.

```
[ "/aws/service/global-infrastructure/regions/af-south-1",
  "/aws/service/global-infrastructure/regions/ap-east-1",
  "/aws/service/global-infrastructure/regions/ap-northeast-3",
  "/aws/service/global-infrastructure/regions/ap-southeast-1",
  "/aws/service/global-infrastructure/regions/ca-central-1",
  "/aws/service/global-infrastructure/regions/cn-north-1",
  "/aws/service/global-infrastructure/regions/eu-west-2",
  "/aws/service/global-infrastructure/regions/eu-west-3",
  "/aws/service/global-infrastructure/regions/us-east-1",
  "/aws/service/global-infrastructure/regions/us-gov-west-1",
  "/aws/service/global-infrastructure/regions/ap-northeast-2",
  "/aws/service/global-infrastructure/regions/ap-south-1",
  "/aws/service/global-infrastructure/regions/ap-southeast-2",
  "/aws/service/global-infrastructure/regions/cn-northwest-1",
  "/aws/service/global-infrastructure/regions/eu-south-1",
  "/aws/service/global-infrastructure/regions/me-south-1",
  "/aws/service/global-infrastructure/regions/na-east-1",
  "/aws/service/global-infrastructure/regions/us-east-2",
  "/aws/service/global-infrastructure/regions/us-gov-east-1",
  "/aws/service/global-infrastructure/regions/us-west-1",
  "/aws/service/global-infrastructure/regions/ap-northeast-1",
  "/aws/service/global-infrastructure/regions/eu-central-1",
  "/aws/service/global-infrastructure/regions/eu-north-1",
  "/aws/service/global-infrastructure/regions/us-west-1",
  "/aws/service/global-infrastructure/regions/us-west-2"
]
```

View available AWS services

You can view a complete list of all available AWS services and sort them into alphabetical order by using the following command. This example output has been truncated for space.

Linux

```bash
aws ssm get-parameters-by-path
   --path /aws/service/global-infrastructure/services
   --query 'Parameters[].Name | sort(@)'
```
Windows

```bash
aws ssm get-parameters-by-path ^
    --path /aws/service/global-infrastructure/services ^
    --query "Parameters[].Name | sort(@)"
```

The command returns information like the following.

```
[  
  "/aws/service/global-infrastructure/services/acm-pca",  
  "/aws/service/global-infrastructure/services/acm",  
  "/aws/service/global-infrastructure/services/alexaforbusiness",  
  "/aws/service/global-infrastructure/services/apigateway",  
  "/aws/service/global-infrastructure/services/application-autoscaling",  
  "/aws/service/global-infrastructure/services/appmesh",  
  "/aws/service/global-infrastructure/services/appstream",  
  "/aws/service/global-infrastructure/services/appsync",  
  "/aws/service/global-infrastructure/services/athena",  
  "/aws/service/global-infrastructure/services/autoscaling-plans",  
  "/aws/service/global-infrastructure/services/autoscaling",  
  "/aws/service/global-infrastructure/services/backup",  
  "/aws/service/global-infrastructure/services/batch",  
  "/aws/service/global-infrastructure/services/budgets",  
  "/aws/service/global-infrastructure/services/ce",  
  "/aws/service/global-infrastructure/services/chime",  
  "/aws/service/global-infrastructure/services/cloud9",  
  "/aws/service/global-infrastructure/services/cloudformation",  
  "/aws/service/global-infrastructure/services/cloudfront",  
  "/aws/service/global-infrastructure/services/cloudhsm",  
  "/aws/service/global-infrastructure/services/cloudhsmv2",  
  "/aws/service/global-infrastructure/services/cloudsearch",  
  "/aws/service/global-infrastructure/services/cloudtrail",  
  "/aws/service/global-infrastructure/services/cloudwatch",  
  "/aws/service/global-infrastructure/services/codebuild" ,
]
```

View supported Regions for an AWS service

You can view a list of AWS Regions where a service is available. This example uses Systems Manager (ssm).

Linux

```bash
aws ssm get-parameters-by-path \  
    --path /aws/service/global-infrastructure/services/ssm/regions \  
    --query 'Parameters[].Value'
```

Windows

```bash
aws ssm get-parameters-by-path ^
    --path /aws/service/global-infrastructure/services/ssm/regions ^
    --query Parameters[].Value
```

The command returns information like the following.

```
[  
  "ap-south-1",  
  "ca-central-1",  
  "cn-north-1",  
]
```
Public parameters

"eu-central-1",
"eu-west-1",
"eu-west-2",
"eu-west-3",
"me-south-1",
"us-east-2",
"us-gov-west-1",
"af-south-1",
"ap-east-1",
"ap-northeast-1",
"ap-northeast-2",
"ap-southeast-1",
"ap-southeast-2",
"eu-north-1",
"eu-south-1",
"us-gov-east-1",
"us-west-1",
"cn-northwest-1",
"sa-east-1",
"us-east-1",
"us-west-2"

View the regional endpoint for a service

You can view a regional endpoint for a service by using the following command.

Linux

```bash
aws ssm get-parameter \
  --name /aws/service/global-infrastructure/regions/us-west-1/services/ssm/endpoint \
  --query 'Parameter.Value'
```

Windows

```bash
aws ssm get-parameter ^
  --name /aws/service/global-infrastructure/regions/us-west-1/services/ssm/endpoint ^
  --query Parameter.Value
```

The command returns information like the following.

"ssm.us-west-1.amazonaws.com"

View complete availability zone details

You can view availability zones by using the following command.

Linux

```bash
aws ssm get-parameters-by-path \
  --path /aws/service/global-infrastructure/availability-zones/
```

Windows

```bash
aws ssm get-parameters-by-path ^
  --path /aws/service/global-infrastructure/availability-zones/
```

The command returns information like the following. This example has been truncated for space.
Public parameters

```json
{
    "Parameters": [
        {
            "Name": "/aws/service/global-infrastructure/availability-zones/apne3-az3",
            "Type": "String",
            "Value": "apne3-az3",
            "Version": 1,
            "LastModifiedDate": "2020-04-03T14:29:28.995000-07:00",
            "ARN": "arn:aws:ssm:us-west-2::parameter/aws/service/global-infrastructure/availability-zones/apne3-az3"
        },
        {
            "Name": "/aws/service/global-infrastructure/availability-zones/aps1-az2",
            "Type": "String",
            "Value": "aps1-az2",
            "Version": 1,
            "LastModifiedDate": "2020-04-03T14:21:02.690000-07:00",
        },
        {
            "Name": "/aws/service/global-infrastructure/availability-zones/cnn1-az2",
            "Type": "String",
            "Value": "cnn1-az2",
            "Version": 1,
            "LastModifiedDate": "2020-04-03T14:19:57.254000-07:00",
            "ARN": "arn:aws:ssm:us-west-2::parameter/aws/service/global-infrastructure/availability-zones/cnn1-az2"
        }
    ]
}
```

View availability zone names only

You can view the names of availability zones only by using the following command.

**Linux**

```
aws ssm get-parameters-by-path
   --path /aws/service/global-infrastructure/availability-zones
   --query 'Parameters[].Name | sort(@)
```

**Windows**

```
aws ssm get-parameters-by-path
   --path /aws/service/global-infrastructure/availability-zones
   --query "Parameters[].Name | sort(@)"
```

The command returns information like the following. This example has been truncated for space.

```
["/aws/service/global-infrastructure/availability-zones/afs1-az1",
 "/aws/service/global-infrastructure/availability-zones/afs1-az2",
 "/aws/service/global-infrastructure/availability-zones/afs1-az3",
 "/aws/service/global-infrastructure/availability-zones/apne1-az1",
 "/aws/service/global-infrastructure/availability-zones/apne1-az2",
 "/aws/service/global-infrastructure/availability-zones/apne1-az3",
 "/aws/service/global-infrastructure/availability-zones/apne1-az4",
 "/aws/service/global-infrastructure/availability-zones/apne3-az1",
 "/aws/service/global-infrastructure/availability-zones/apne3-az2",
 "/aws/service/global-infrastructure/availability-zones/apne3-az3",
 "/aws/service/global-infrastructure/availability-zones/apne3-az4",
"]```
View names of availability zones in a single Region

You can view the names of the availability zones in one Region (us-east-1, in this example) using the following command.

Linux

```
aws ssm get-parameters-by-path \
   --path /aws/service/global-infrastructure/regions/us-east-1/availability-zones \
   --query 'Parameters[].Name | sort(@)
```

Windows

```
aws ssm get-parameters-by-path ^
   --path /aws/service/global-infrastructure/regions/us-east-1/availability-zones ^
   --query "Parameters[].Name | sort(@)
```

The command returns information like the following. This example has been truncated for space.

```
[ 
  "/aws/service/global-infrastructure/regions/us-east-1/availability-zones/use1-az1",
  "/aws/service/global-infrastructure/regions/us-east-1/availability-zones/use1-az2",
  "/aws/service/global-infrastructure/regions/us-east-1/availability-zones/use1-az3",
  "/aws/service/global-infrastructure/regions/us-east-1/availability-zones/use1-az4",
  "/aws/service/global-infrastructure/regions/us-east-1/availability-zones/use1-az5",
  "/aws/service/global-infrastructure/regions/us-east-1/availability-zones/use1-az6"
]
```

View availability zone ARNs only

You can view the ARNs of availability zones only by using the following command.

Linux

```
aws ssm get-parameters-by-path \
   --path /aws/service/global-infrastructure/availability-zones \
   --query 'Parameters[].ARN | sort(@)
```

Windows

```
aws ssm get-parameters-by-path ^
   --path /aws/service/global-infrastructure/availability-zones ^
   --query "Parameters[].ARN | sort(@)
```

The command returns information like the following.

```
[ 
]
```
Public parameters


View local zone details

You can view local zones by using the following command.

**Note**
Only one local zone is currently available.

Linux

```bash
aws ssm get-parameters-by-path \
   --path /aws/service/global-infrastructure/local-zones/
```

Windows

```bash
aws ssm get-parameters-by-path ^ 
   --path /aws/service/global-infrastructure/local-zones/
```

The command returns information like the following.

```json
{
   "Parameters": [
       {
           "Name": "/aws/service/global-infrastructure/local-zones/usw2-lax1-az1",
           "Type": "String",
           "Value": "usw2-lax1-az1",
           "Version": 1,
           "LastModifiedDateTime": "2020-04-29T20:27:03.291000-07:00",
           "ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/usw2-lax1-az1"
       }
   ]
}
```

View all parameters and values under a local zone

You can view all parameter data for a local zone by using the following command.

Linux

```bash
aws ssm get-parameters-by-path \
   --path "/aws/service/global-infrastructure/local-zones/usw2-lax1-az1/"
```

Windows

```bash
aws ssm get-parameters-by-path ^ 
   --path "/aws/service/global-infrastructure/local-zones/usw2-lax1-az1/"
```

The command returns information like the following.

```json
{
   "Parameters": [ 
       {

```
"Name": "aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/geolocationCountry",
"Type": "String",
"Value": "US",
"Version": 1,
"LastModifiedDate": "2020-04-29T20:03:430000-07:00",
},
{
"Name": "aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/geolocationRegion",
"Type": "String",
"Value": "US-CA",
"Version": 1,
"LastModifiedDate": "2020-04-29T20:03:489000-07:00",
"ARN": "arn:aws:ssm:us-east-2::parameter/aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/geolocationRegion"
},
{
"Name": "aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/network-border-group",
"Type": "String",
"Value": "us-west-2-lax-1",
"Version": 1,
"LastModifiedDate": "2020-04-29T20:03:611000-07:00",
},
{
"Name": "aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/parent-availability-zone",
"Type": "String",
"Value": "usw2-az2",
"Version": 1,
"LastModifiedDate": "2020-04-29T20:03:736000-07:00",
},
{
"Name": "aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/parent-region",
"Type": "String",
"Value": "us-west-2",
"Version": 1,
"LastModifiedDate": "2020-04-29T20:03:670000-07:00",
},
{
"Name": "aws/service/global-infrastructure/local-zones/usw2-lax-1-az1/zone-group",
"Type": "String",
"Value": "us-west-2-lax-1",
"Version": 1,
"LastModifiedDate": "2020-04-29T20:03:552000-07:00",
}]

View local zone parameter names only

You can view just the names of local zone parameters by using the following command.
Getting started with Parameter Store

Before setting up parameters in Systems Manager Parameter Store, you first configure AWS Identity and Access Management (IAM) policies that provide users in your account with permission to perform the actions you specify.

This section includes information about how to manually configure these policies using the IAM console, and how to assign them to users and user groups. You can also create and assign policies to control which parameter actions can be run on an instance.

This section also include information about how to create Amazon CloudWatch Events rules that let you receive notifications about changes to Systems Manager parameters. You can also use CloudWatch Events rules to trigger other actions in AWS based on changes in Parameter Store.

Contents

- Restricting access to Systems Manager parameters using IAM policies (p. 224)
- Setting up notifications or trigger actions based on Parameter Store events (p. 226)
- Managing parameter tiers (p. 230)
- Increasing Parameter Store throughput (p. 236)

Restricting access to Systems Manager parameters using IAM policies

You restrict access to Systems Manager parameters by using AWS Identity and Access Management (IAM). More specifically, you create IAM policies that restrict access to the following API operations:
When using IAM policies to restrict access to Systems Manager parameters, we recommend that create and use restrictive IAM policies. For example, the following policy allows a user to call the DescribeParameters and GetParameters API operations for a limited set of resources. This means that the user can get information about and use all parameters that begin with prod-.*.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": ["ssm:DescribeParameters"],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": ["ssm:GetParameters"],
      }
   ]
}
```

For trusted administrators, you can provide access to all Systems Manager parameter API operations by using a policy similar to the following example. This policy gives the user full access to all production parameters that begin with dbserver-prod-.*.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
      },
      {
         "Sid": "VisualEditor1",
         "Effect": "Allow",
         "Action": "ssm:DescribeParameters",
      }
   ]
}
```
Allowing only specific parameters to run on instances

You can control access so that instances can run only parameters that you specify.

If you choose the SecureString parameter type when you create your parameter, Systems Manager uses AWS Key Management Service (KMS) to encrypt the parameter value. AWS KMS encrypts the value by using either an AWS-managed customer master key (CMK) or a customer managed CMK. For more information about AWS KMS and CMKs, see the AWS Key Management Service Developer Guide.

You can view the AWS-managed CMK by running the following command from the AWS CLI:

```bash
aws kms describe-key --key-id alias/aws/ssm
```

The following example enables instances to get a parameter value only for parameters that begin with "prod-" if the parameter is a SecureString parameter, then the instance decrypts the string using AWS KMS.

**Note**

Instance policies, like in the following example, are assigned to the instance role in IAM. For more information about configuring access to Systems Manager features, including how to assign policies to users and instances, see Setting up AWS Systems Manager (p. 25).

```json
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect": "Allow",
         "Action": ["ssm:GetParameters"],
      },
      {
         "Effect": "Allow",
         "Action": ["kms:Decrypt"],
         "Resource": ["arn:aws:kms:region:account-id:key/CMK"]
      }
   ]
}
```

Setting up notifications or trigger actions based on Parameter Store events

The topics in this section explain how to use Amazon CloudWatch Events and Amazon Simple Notification Service (Amazon SNS) to notify you about changes to Systems Manager parameters. You
can create a CloudWatch rule to notify you when a parameter or a parameter label version is created, updated, or deleted. You can be notified about changes or status related to parameter policies, such as when a parameter expires, is going to expire, or hasn’t changed for a specified period of time.

**Note**
Parameter policies are available for parameters that use the advanced parameters tier. Charges apply. For more information, see Assigning parameter policies (p. 242) and Managing parameter tiers (p. 230).

The topics below also explain how to trigger other actions on a target for specific parameter events. For example, you can run an AWS Lambda function to recreate a parameter automatically when it expires or is deleted. You can set up a notification to trigger a Lambda function when your database password is updated. The Lambda function can force your database connections to reset or reconnect with the new password. CloudWatch Events also supports running Run Command commands and Automations executions, and actions in many other AWS services. For more information, see the Amazon CloudWatch Events User Guide.

**Before You Begin**
Create any resources you need to specify the target action for the rule you create. For example, if the rule you create is for sending a notification, first create an Amazon SNS topic. For more information, see Getting Started with Amazon SNS in the Amazon Simple Notification Service Developer Guide.

**Topics**
- Configuring CloudWatch Events for parameters (p. 227)
- Configuring CloudWatch Events for parameter policies (p. 228)

**Configuring CloudWatch Events for parameters**
This topic explains how to create a CloudWatch Events rule that invokes a target based on events that happen to one or more parameters in your AWS account.

**To configure CloudWatch Events for Systems Manager parameters**
1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Events**, and then choose **Create rule**.
3. Under **Event Source**, verify that **Event Pattern** is selected.
4. Above the **Event Pattern Preview** field, choose **Edit**.
   **Note**
   You are modifying sample code we provide instead of using the event pattern builder fields.
5. Replace the content in the edit box with the following:

```json
{
    "source": ["aws.ssm"],
    "detail-type": ["Parameter Store Change"],
    "detail": {
        "name": [
            "parameter-1-name",
            "/parameter-2-name/level-2",
            "/parameter-3-name/level-2/level-3"
        ]
    }
}
```
6. Modify the contents for the parameters and the operations you want to take action on.

For example, the following content means an action is taken when either of the parameters named /Oncall and /Project/Teamlead are updated:

```json
{
    "source": [
        "aws.ssm"
    ],
    "detail-type": [
        "Parameter Store Change"
    ],
    "detail": {
        "name": [
            "/Oncall",
            "/Project/Teamlead"
        ],
        "operation": [
            "Update"
        ]
    }
}
```

7. Choose Save.

8. For Targets, choose Add targets.

9. In the Targets list, choose a target type. For example, choose Lambda function or SNS topic.

10. Expand Configure input and choose an option. Then provide any other configuration details required by the target type you selected.

11. Scroll to the bottom of the page, if necessary, and then choose Configure details.

12. Provide a name and (optional) description for the CloudWatch Events rule. Leave the Enabled box selected to make the rule active immediately.

13. Choose Create rule.

### Configuring CloudWatch Events for parameter policies

This topic explains how to create CloudWatch Events rules that invoke targets based on events that happen to one or more parameter policies in your AWS account. When you create an advanced parameter, you specify when a parameter expires, when to receive notification before a parameter expires, and how long to wait before notification should be sent that a parameter hasn’t changed. You set up notification for these events using the following procedure. For more information, see Assigning parameter policies (p. 242) and Managing parameter tiers (p. 230).

**To configure CloudWatch Events for Systems Manager parameter policies**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Under Event Source, verify that Event Pattern is selected.
4. Above the Event Pattern Preview field, choose Edit.

   **Note**
   You are modifying sample code we provide instead of using the event pattern builder fields.

5. Replace the content in the edit box with the following:

   ```json
   { 
     "source": [ 
       "aws.ssm"
     ],
     "detail-type": [ 
       "Parameter Store Policy Action"
     ],
     "detail": { 
       "parameter-name": [ 
         "parameter-1-name",
         "/parameter-2-name/level-2",
         "/parameter-3-name/level-2/level-3"
       ],
       "policy-type": [ 
         "Expiration",
         "ExpirationNotification",
         "NoChangeNotification"
       ]
     }
   }
   ```

6. Modify the contents for the parameters and the policy types you want to take action on.
   For example, the following content means an action is taken whenever the parameter named /OncallDuties expires and is deleted:

   ```json
   { 
     "source": [ 
       "aws.ssm"
     ],
     "detail-type": [ 
       "Parameter Store Policy Action"
     ],
     "detail": { 
       "parameter-name": [ 
         "/OncallDuties"
       ],
       "policy-type": [ 
         "Expiration"
       ]
     }
   }
   ```

7. Choose Save.

8. For Targets, choose Add targets.

9. In the Targets list, choose a target type. For example, choose Lambda function or SNS topic.

10. Expand Configure input and choose an option. Then provide any other configuration details required by the target type you selected.

11. Scroll to the bottom of the page, if necessary, and then choose Configure details.

12. Provide a name and (optional) description for the CloudWatch Events rule. Leave the Enabled box selected to make the rule active immediately.

13. Choose Create rule.

**Related Information**
Managing parameter tiers

AWS Systems Manager Parameter Store includes standard parameters and advanced parameters. You individually configure parameters to use either the standard-parameter tier (the default tier) or the advanced-parameter tier.

You can change a standard parameter to an advanced parameter at any time, but you can't revert an advanced parameter to a standard parameter. Reverting an advanced parameter to a standard parameter would result in data loss because the system would truncate the size of the parameter from 8 KB to 4 KB. Reverting would also remove any policies attached to the parameter. Also, advanced parameters use a different form of encryption than standard parameters. For more information, see How AWS Systems Manager Parameter Store Uses AWS KMS in the AWS Key Management Service Developer Guide.

If you no longer need an advanced parameter, or if you no longer want to incur charges for an advanced parameter, you must delete it and recreate it as a new standard parameter.

The following table describes the differences between the tiers.

<table>
<thead>
<tr>
<th></th>
<th>Standard</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of parameters allowed (per AWS account and Region)</td>
<td>10,000</td>
<td>100,000</td>
</tr>
<tr>
<td>Maximum size of a parameter value</td>
<td>4 KB</td>
<td>8 KB</td>
</tr>
<tr>
<td>Parameter policies available</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>For more information, see Assigning parameter policies (p. 242).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>No additional charge</td>
<td>Charges apply</td>
</tr>
<tr>
<td>For more information, see AWS Systems Manager Pricing.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Topics

• Specifying a default parameter tier (p. 230)
• Changing a standard parameter to an advanced parameter (p. 236)

Specifying a default parameter tier

In requests to create or update a parameter (that is, the PutParameter action), you can specify the parameter tier to use in the request. The following is an example, using the AWS CLI.
Whenever you specify a tier in the request, Parameter Store creates or updates the parameter according to your request. However, if you do not explicitly specify a tier in a request, the Parameter Store default tier setting determines which tier the parameter is created in.

The default tier when you begin using Parameter Store is the standard-parameter tier. If you use the advanced-parameter tier, you can specify one of the following as the default:

- **Advanced**: With this option, Parameter Store evaluates all requests as advanced parameters.
- **Intelligent-Tiering**: With this option, Parameter Store evaluates each request to determine if the parameter is standard or advanced.

If the request doesn't include any options that require an advanced parameter, the parameter is created in the standard-parameter tier. If one or more options requiring an advanced parameter are included in the request, Parameter Store create a parameter in the advanced-parameter tier.

**Benefits of Intelligent-Tiering**

The following are reasons you might choose Intelligent-Tiering as the default tier.

- **Cost control** – Intelligent-Tiering helps control your parameter-related costs by always creating standard parameters unless an advanced parameter is absolutely necessary.

- **Automatic upgrade to the advanced-parameter tier** – When you make a change to your code that requires upgrading a standard parameter to an advanced parameter, Intelligent-Tiering handles the conversion for you. You do not need to change your code to handle the upgrade.

Here are some examples of automatic upgrade:

- Your AWS CloudFormation templates provision numerous parameters when they are run. When this process causes you to reach the 10,000 parameter limit in the standard-parameter tier, Intelligent-Tiering automatically upgrades you to the advanced-parameter tier, and your AWS CloudFormation processes are not interrupted.

- You store a certificate value in a parameter, rotate the certificate value regularly, and the content is less than the 4 KB limit of the standard-parameter tier. If a replacement certificate value exceeds 4 KB, Intelligent-Tiering automatically upgrades the parameter to the advanced-parameter tier.

- You want to associate numerous existing standard parameters to a parameter policy, which requires the advanced-parameter tier. Instead of your having to include the option `--tier Advanced` in all of the calls to update the parameters, Intelligent-Tiering automatically upgrades the parameters to the advanced-parameter tier. The Intelligent-Tiering option upgrades parameters from standard to advanced whenever criteria for the advanced-parameter tier are introduced.

Options that require an advanced parameter include the following:

- The content size of the parameter is more than 4 KB.
- The parameter uses a parameter policy.
- More than 10,000 parameters already exist in your AWS account in the current Region.

**Default Tier Options**
The tier options you can specify as the default include the following.

- **Standard** – The standard-parameter tier is the default tier when you begin to use Parameter Store. Using the standard-parameter tier, you can create 10,000 parameters for each Region in an AWS account. The content size of each parameter can equal a maximum of 4 KB. Standard parameters do not support parameter policies. There is no additional charge to use the standard-parameter tier. Choosing **Standard** as the default tier means that Parameter Store always attempts to create a standard parameter for requests that don't specify a tier.

- **Advanced** – The advanced-parameter tier lets you create a maximum of 100,000 parameters for each Region in an AWS account. The content size of each parameter can equal a maximum of 8 KB. Advanced parameters support parameter policies. There is a charge to use the advanced-parameter tier. For more information, see AWS Systems Manager Pricing. Choosing **Advanced** as the default tier means that Parameter Store always attempts to create an advanced parameter for requests that don't specify a tier.

  **Note**
  When you choose the advanced-parameter tier, you must explicitly authorize AWS to charge your account for any advanced parameters you create.

- **Intelligent-Tiering** – The Intelligent-Tiering option lets Parameter Store determine whether to use the standard-parameter tier or advanced-parameter tier based on the content of the request. For example, if you run a command to create a parameter with content under 4 KB, and there are fewer than 10,000 parameters in the current Region in your AWS account, and you do not specify a parameter policy, a standard parameter is created. If you run a command to create a parameter with more than 4 KB of content, you already have more than 10,000 parameters in the current Region in your AWS account, or you specify a parameter policy, an advanced parameter is created.

  **Note**
  When you choose Intelligent-Tiering, you must explicitly authorize AWS to charge your account for any advanced parameters that are created.

You can change the Parameter Store default tier setting at any time.

**Configuring permissions to specify a Parameter Store default tier**

Verify that you have permission in AWS Identity and Access Management (IAM) to change the default parameter tier in Parameter Store by doing one of the following:

- Ensure that the AdministratorAccess policy is attached to your IAM user, group, or role.
- Ensure that you have permission to change the default tier setting by using the following API actions:

  - GetServiceSetting
  - UpdateServiceSetting
  - ResetServiceSetting

Use the following procedure to add an inline IAM policy to a user account. This policy enables a user to view and change the default tier setting for parameters in a specific Region in an AWS account.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**.
3. In the list, choose the name of the user to attach a policy to.
4. Choose the **Permissions** tab.
5. On the right side of the page, under **Permission policies**, choose **Add inline policy**.
6. Choose the **JSON** tab.
7. Replace the default content with the following:
8. Choose **Review policy**.
9. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **Parameter-Store-Default-Tier** or another name you prefer.
10. Choose **Create policy**.

Administrators can specify read-only permission by assigning the following inline policy to the user's account.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:GetServiceSetting"
            ],
            "Resource": "*
        },
        {
            "Effect": "Deny",
            "Action": [
                "ssm:ResetServiceSetting",
                "ssm:UpdateServiceSetting"
            ],
        }
    ]
}
```

For more information about creating and editing IAM policies, see **Creating IAM Policies** in the **IAM User Guide**.

**Specifying or changing the Parameter Store default tier (console)**

The following procedure shows how to use the Systems Manager console to specify or change the default parameter tier for the current AWS account and Region.

**Tip**
If you haven't created a parameter yet, you can use the AWS CLI or Tools for Windows PowerShell to change the default parameter tier. For information, see **Specifying or changing**
To specify or change the Parameter Store default tier

2. In the navigation pane, choose Parameter Store.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Parameter Store.
3. Choose the Settings tab.
4. Choose Set default.
5. Choose one of the following options.
   - Standard
   - Advanced
   - Intelligent-Tiering

   For information about these options, see Specifying a default parameter tier (p. 230).
6. Review the message, and choose Confirm.

If you want to change the default tier setting later, repeat this procedure and specify a different default tier option.

Specifying or changing the Parameter Store default tier (AWS CLI)

The following procedure shows how to use the AWS Command Line Interface (AWS CLI) to change the default parameter tier setting for the current AWS account and Region.

To specify or change the Parameter Store default tier using the AWS CLI

1. Open the AWS CLI and run the following command to change the default parameter tier setting for a specific Region in an AWS account.

   ```
   ```

   *region* represents the identifier for an AWS Region supported by AWS Systems Manager, such as us-east-2 for the US East (Ohio) Region. For a list of supported *region* values, see the *Region* column in Systems Manager service endpoints in the Amazon Web Services General Reference.

   *tier-option* values include Standard, Advanced, and Intelligent-Tiering. For information about these options, see Specifying a default parameter tier (p. 230).

   There is no output if the command succeeds.

2. Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and Region.

   ```
   ```

   The system returns information similar to the following:
If you want to change the default tier setting again, repeat this procedure and specify a different `SettingValue` option.

**Specifying or changing the Parameter Store default tier (PowerShell)**

The following procedure shows how to use the AWS Tools for Windows PowerShell to change the default parameter tier setting for a specific Region in an AWS account.

**To specify or change the Parameter Store default tier using PowerShell**

1. Change the Parameter Store default tier in the current AWS account and Region using the AWS Tools for PowerShell.

   ```powershell
   ```

   `region` represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported `region` values, see the `Region` column in Systems Manager service endpoints in the Amazon Web Services General Reference.

   `tier-option` values include Standard, Advanced, and Intelligent-Tiering. For information about these options, see Specifying a default parameter tier (p. 230).

   There is no output if the command succeeds.

2. Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and Region.

   ```powershell
   ```

   `region` represents the identifier for an AWS Region supported by AWS Systems Manager, such as `us-east-2` for the US East (Ohio) Region. For a list of supported `region` values, see the `Region` column in Systems Manager service endpoints in the Amazon Web Services General Reference.

   The system returns information similar to the following:

   ```
   LastModifiedDate : 4/29/2019 3:35:44 PM
   LastModifiedUser : arn:aws:sts::123456789012:assumed-role/Administrator/Jasper
   SettingId        : /ssm/parameter-store/default-parameter-tier
   SettingValue     : Advanced
   Status           : Customized
   ```
If you want to change the default tier setting again, repeat this procedure and specify a different `SettingValue` option.

**Changing a standard parameter to an advanced parameter**

Use the following procedure to change an existing standard parameter to an advanced parameter. For information about how to create a new advanced parameter, see Creating Systems Manager parameters (p. 249).

**To change a standard parameter to an advanced parameter**

2. In the navigation pane, choose **Parameter Store**.
3. Choose a parameter, and then choose **Edit**.
4. For **Description**, enter information about this parameter.
5. Choose **Advanced**.
6. For **Value**, enter the value of this parameter. Advanced parameters have a maximum value limit of 8 KB.
7. Choose **Save changes**.

**Increasing Parameter Store throughput**

Increasing Parameter Store throughput increases the maximum number of transactions per second (TPS) that Parameter Store can process. Increased throughput enables you to operate Parameter Store at higher volumes to support applications and workloads that need concurrent access to a large number of parameters. You can increase the limit to 1,000 TPS on the **Settings** tab. Increasing the throughput limit incurs a charge on your AWS account. For more information, see AWS Systems Manager Pricing.

**Note**

The Parameter Store throughput setting applies to all transactions created by all AWS Identity and Access Management (IAM) users in the current AWS account and Region. The throughput setting applies to standard and advanced parameters.

**Configuring permissions to increase Parameter Store throughput**

Verify that you have permission in AWS Identity and Access Management (IAM) to increase Parameter Store throughput by doing one of the following:

- Ensure that the **AdministratorAccess** policy is attached to your IAM user, group, or role.
- Ensure that you have permission to change the throughput service setting by using the following API actions:
  - `GetServiceSetting`
  - `UpdateServiceSetting`
  - `ResetServiceSetting`

Use the following procedure to add an inline IAM policy to a user account. This policy enables a user to view and change the parameter-throughput setting for parameters in their account and Region.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Users**.
3. In the list, choose the name of the user to attach a policy to.
4. Choose the **Permissions** tab.
5. On the right side of the page, under **Permission policies**, choose **Add inline policy**.
6. Choose the **JSON** tab.
7. Replace the default content with the following:

```json
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Effect": "Allow",
      "Action": [
         "ssm:GetServiceSetting"
      ],
      "Resource": "*
   },
   {
      "Effect": "Allow",
      "Action": [
         "ssm:UpdateServiceSetting"
      ],
   }
]
}
```

8. Choose **Review policy**.
9. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **Parameter-Store-Throughput** or another name you prefer.
10. Choose **Create policy**.

Administrators can specify read-only permission by assigning the following inline policy to the user’s account.

```json
{
   "Version": "2012-10-17",
   "Statement": [
   {
      "Effect": "Allow",
      "Action": [
         "ssm:GetServiceSetting"
      ],
      "Resource": "**"
   },
   {{
      "Effect": "Deny",
      "Action": [
         "ssm:ResetServiceSetting",
         "ssm:UpdateServiceSetting"
      ],
      "Resource": "**"
   }
]
}
```

For more information about creating and editing IAM policies, see Creating IAM Policies in the IAM User Guide.

**Increasing throughput (console)**

The following procedure shows how to use the Systems Manager console to increase the number of transactions per second that Parameter Store can process for the current AWS account and Region.
Tip
If you haven’t created a parameter yet, you can use the AWS CLI or AWS Tools for Windows PowerShell to increase throughput. For information, see Increasing throughput (AWS CLI) (p. 238) and Increasing throughput (PowerShell) (p. 239).

To increase Parameter Store throughput
2. In the navigation pane, choose Parameter Store.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Parameter Store.
3. Choose the Settings tab.
4. Choose Set limit.
5. Review the message, and choose Accept.

If you no longer need increased throughput, or if you no longer want to incur charges, you can revert to the standard settings. To revert your settings, repeat this procedure and choose Reset limit.

Increasing throughput (AWS CLI)
The following procedure shows how to use the AWS Command Line Interface (AWS CLI) to increase the number of transactions per second that Parameter Store can process for the current AWS account and Region.

To increase Parameter Store throughput using the AWS CLI
1. Open the AWS CLI and run the following command to increase the transactions per second that Parameter Store can process in the current AWS account and Region.

   ```shell
   aws ssm update-service-setting --setting-id arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled --setting-value true
   ```

   There is no output if the command succeeds.

2. Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and Region.

   ```shell
   ```

   ```json
   ```
If you no longer need increased throughput, or if you no longer want to incur charges, you can revert to the standard settings. To revert your settings, run the following command.

```
```

```json
{
    "ServiceSetting": {
        "SettingId": "/ssm/parameter-store/high-throughput-enabled",
        "SettingValue": "false",
        "LastModifiedDate": 1555532818.578,
        "LastModifiedUser": "System",
        "Status": "Default"
    }
}
```

### Increasing throughput (PowerShell)

The following procedure shows how to use the AWS Tools for Windows PowerShell to increase the number of transactions per second that Parameter Store can process for the current AWS account and Region.

#### To increase Parameter Store throughput using PowerShell

1. Increase Parameter Store throughput in the current AWS account and Region using the AWS Tools for PowerShell.

   ```powershell
   ```

   There is no output if the command succeeds.

2. Run the following command to view the current throughput service settings for Parameter Store in the current AWS account and Region.

   ```powershell
   ```

<table>
<thead>
<tr>
<th>ARN</th>
<th>LastModifiedDate</th>
<th>LastModifiedUser</th>
<th>SettingId</th>
<th>SettingValue</th>
<th>Status</th>
</tr>
</thead>
</table>

If you no longer need increased throughput, or if you no longer want to incur charges, you can revert to the standard settings. To revert your settings, run the following command.

```
Reset-SSMServiceSetting -SettingId "arn:aws:ssm:region:account-id:servicesetting/ssm/parameter-store/high-throughput-enabled" -Region region
```

```powershell
```
Working with parameters

This section describes how to organize and create tag parameters, and how to create different versions of parameters.

Topics

- Organizing parameters into hierarchies (p. 240)
- Assigning parameter policies (p. 242)
- About requirements and constraints for parameter names (p. 248)
- Creating Systems Manager parameters (p. 249)
- Searching for Systems Manager parameters (p. 257)
- Working with parameter versions (p. 259)
- Labeling parameters (p. 260)

Organizing parameters into hierarchies

Managing dozens or hundreds of parameters as a flat list is time consuming and prone to errors. It can also be difficult to identify the correct parameter for a task. This means you might accidentally use the wrong parameter, or you might create multiple parameters that use the same configuration data.

You can use parameter hierarchies to help you organize and manage parameters. A hierarchy is a parameter name that includes a path that you define by using forward slashes (/).

Topics

- Parameter hierarchy examples (p. 240)
- Querying parameters in a hierarchy (p. 241)
- Restricting access to Parameter Store API actions (p. 241)

Parameter hierarchy examples

The following example uses three hierarchy levels in the name to identify the following:

/Environment/Type of computer/Application/Data
/Dev/DBServer/MySQL/db-string13

You can create a hierarchy with a maximum of 15 levels. We suggest that you create hierarchies that reflect an existing hierarchical structure in your environment, as shown in the following examples:

- Your Continuous integration and Continuous delivery environment (CI/CD workflows)
  
  /Dev/DBServer/MySQL/db-string
  /Staging/DBServer/MySQL/db-string
  /Prod/DBServer/MySQL/db-string

- Your applications that use containers
Parameter hierarchies standardize the way you create parameters and make it easier to manage parameters over time. A parameter hierarchy can also help you identify the correct parameter for a configuration task. This helps you to avoid creating multiple parameters with the same configuration data.

You can create a hierarchy that allows you to share parameters across different environments, as shown in the following examples that use passwords in development and staging environment.

/DevTest/MyApp/database/my-password

You could then create a unique password for your production environment, as shown in the following example:

/prod/MyApp/database/my-password

You are not required to specify a parameter hierarchy. You can create parameters at level one. These are called root parameters. For backward compatibility, all parameters created in Parameter Store before hierarchies were released are root parameters. The systems treats both of the following parameters as root parameters.

/parameter-name

parameter-name

For an example of how to work with parameter hierarchies, see Walkthrough: Manage parameters using hierarchies (AWS CLI) (p. 274).

### Querying parameters in a hierarchy

Another benefit of using hierarchies is the ability to query for all parameters within a hierarchy by using the GetParametersByPath API action. For example, if you run the following command from the AWS CLI, the system returns all parameters in the IIS level.

```bash
aws ssm get-parameters-by-path --path /Dev/Web/IIS
```

To view decrypted SecureString parameters in a hierarchy, you specify the path and the `--with-decryption` parameter, as shown in the following example.

```bash
aws ssm get-parameters-by-path --path /Prod/ERP/SAP --with-decryption
```

### Restricting access to Parameter Store API actions

Using AWS Identity and Access Management (IAM) policies, you can provide or restrict user access to Parameter Store API actions and content.

In the following sample policy, users are first granted access to run the PutParameter API action on all parameters in the AWS account 123456789012 in the US East (Ohio) Region (us-east-2). But then users are restricted from changing values of existing parameters because the `overwrite` option is explicitly
denied for the `PutParameter` action. In other words, users who are assigned this policy can create parameters, but not make changes to existing parameters.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ssm:PutParameter"],
    },
    {
      "Effect": "Deny",
      "Action": ["ssm:PutParameter"],
      "Condition": {
        "StringEquals": {
          "ssm:Overwrite": ["true"
        }
      },
    }
  ]
}
```

### Assigning parameter policies

Parameter policies help you manage a growing set of parameters by enabling you to assign specific criteria to a parameter such as an expiration date or time to live. Parameter policies are especially helpful in forcing you to update or delete passwords and configuration data stored in Parameter Store. Parameter Store offers the following types of policies: Expiration, ExpirationNotification, and NoChangeNotification. The policies are described in more detail in this section.

Parameter Store enforces parameter policies by using asynchronous, periodic scans. After you create a policy, you don’t need to perform additional actions to enforce the policy. Parameter Store independently performs the action defined by the policy according to the criteria you specified.

**Note**

Parameter policies are available for parameters that use the advanced parameters tier. For more information, see Managing parameter tiers (p. 230).

A parameter policy is a JSON array, as shown in the following table. You can assign a policy when you create a new advanced parameter, or you can apply a policy by updating a parameter. Parameter Store supports the following types of parameter policies.

<table>
<thead>
<tr>
<th>Policy</th>
<th>Details</th>
<th>Examples</th>
</tr>
</thead>
</table>
| Expiration   | This policy deletes the parameter. You can specify a specific date and   | `{ "Type":"Expiration",
      "Version":"1.0",
      "Attributes":{
        "Timestamp":"2018-12-02T21:34:33.000Z"
      }
    } |
      time by using either the ISO_INSTANT format or the ISO_OFFSET_DATE_TIME format. To change when you want the parameter to be deleted, you must update the |
### Policy Details Examples

<table>
<thead>
<tr>
<th>Policy</th>
<th>Details</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy</td>
<td>Updating a parameter does not affect the expiration date or time of the policy attached to it. When the expiration date and time is reached, Parameter Store deletes the parameter.</td>
<td><strong>Note</strong> This example uses the ISO_INSTANT format. You can also specify a date and time by using the ISO_OFFSET_DATE_TIME format. Here is an example: 2019-11-01T22:13:48.87+10:30:00</td>
</tr>
<tr>
<td>ExpirationNotification</td>
<td>This policy triggers an event in Amazon CloudWatch Events that notifies you about the expiration. By using this policy, you can receive notifications before the expiration time is reached, in units of days or hours.</td>
<td>{ &quot;Type&quot;:&quot;ExpirationNotification&quot;,&quot;Version&quot;:&quot;1.0&quot;, &quot;Attributes&quot;:{ &quot;Before&quot;:&quot;15&quot;, &quot;Unit&quot;:&quot;Days&quot; } }</td>
</tr>
<tr>
<td>NoChangeNotification</td>
<td>This policy triggers an event in CloudWatch if a parameter has not been modified for a specified period of time. This policy type is useful when, for example, a password needs to be changed within a period of time. This policy determines when to send a notification by reading the LastModifiedTime attribute of the parameter. If you change or edit a parameter, the system resets the notification time period based on the new value of LastModifiedTime.</td>
<td>{ &quot;Type&quot;:&quot;NoChangeNotification&quot;,&quot;Version&quot;:&quot;1.0&quot;, &quot;Attributes&quot;:{ &quot;After&quot;:&quot;20&quot;, &quot;Unit&quot;:&quot;Days&quot; } }</td>
</tr>
</tbody>
</table>

You can assign multiple policies to a parameter. For example, you can assign Expiration and ExpirationNotification policies so that the system triggers a CloudWatch Events event to notify you about the impending deletion of a parameter. You can assign a maximum of ten (10) policies to a parameter.

The following example shows a PutParameter API request that assigns four policies to a new SecureString parameter named ProdDB3.
### Adding policies to an existing parameter

This section includes information about how to add policies to an existing parameter by using the AWS Systems Manager console, the AWS CLI, and AWS Tools for Windows PowerShell. For information about how to create a new parameter that includes policies, see Creating Systems Manager parameters (p. 249).

**Topics**

- Add policies to an existing parameter (console) (p. 244)
- Add policies to an existing parameter (AWS CLI) (p. 245)
- Add policies to an existing parameter by using the Tools for Windows PowerShell (p. 247)

**Add policies to an existing parameter (console)**

Use the following procedure to add policies to an existing parameter by using the Systems Manager console.
To add policies to an existing parameter

2. In the navigation pane, choose Parameter Store.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Parameter Store.
3. Choose the option next to the parameter that you want to update to include policies, and then choose Edit.
4. Choose Advanced.
5. (Optional) In the Parameter policies section, choose Enabled. You can specify an expiration date and one or more notification policies for this parameter.
6. Choose Save changes.

Important

- Parameter Store preserves policies on a parameter until you either overwrite the policies with new policies or remove the policies.
- To remove all policies from an existing parameter, edit the parameter and apply an empty policy by using brackets and curly braces, as follows: [{}]
- If you add a new policy to a parameter that already has policies, then Systems Manager overwrites the policies attached to the parameter. The existing policies are deleted. If you want to add a new policy to a parameter that already has one or more policies, then you must copy and paste the original policies, type the new policy, and then save your changes.

Add policies to an existing parameter (AWS CLI)

Use the following procedure to add policies to an existing parameter by using the AWS CLI.

To add policies to an existing parameter

1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Run the following command to add policies to an existing parameter.
   Linux
   ```bash
   aws ssm put-parameter
   --name "parameter-name" \
   --value 'parameter-value' \
   --type parameter-type \
   --overwrite \
   --policies "[{{policies-enclosed-in-brackets-and-curly-braces}}]"
   ```
   Windows
   ```bash
   aws ssm put-parameter
   --name "parameter-name" ^
   --value 'parameter-value' ^
   --type parameter-type ^
   --overwrite ^
   --policies "[{{policies-enclosed-in-brackets-and-curly-braces}}]"
   ```
Here is an example that includes an expiration policy that deletes the parameter after 15 days. The example also includes a notification policy that generates a CloudWatch Events event five (5) days before the parameter is deleted. Last, it includes a NoChangeNotification policy if no changes are made to this parameter after 60 days. The example uses an obfuscated name (3l3vat3131) for a password and an AWS Key Management Service (AWS KMS) customer master key (CMK). For more information about CMKs, see AWS Key Management Service Concepts in the AWS Key Management Service Developer Guide.

Linux

```bash
aws ssm put-parameter
  --name "/Finance/Payroll/3l3vat3131" \
  --value "P@sSwW)rd" \
  --type "SecureString" \
  --overwrite \
  --policies "[{"Type":"Expiration","Version":"1.0","Attributes":{{"Timestamp":"2020-05-13T00:00:00Z"}},{"Type":"ExpirationNotification","Version":"1.0","Attributes":{{"Before":"5","Unit":"Days"}},{"Type":"NoChangeNotification","Version":"1.0","Attributes":{{"After":"60","Unit":"Days"}}}]]"
```

Windows

```bash
aws ssm put-parameter ^ 
  --name "/Finance/Payroll/3l3vat3131" ^ 
  --value "P@sSwW)rd" ^ 
  --type "SecureString" ^ 
  --overwrite ^ 
  --policies "[{"Type":"Expiration","Version":"1.0","Attributes":{{"Timestamp":"2020-05-13T00:00:00Z"}},{"Type":"ExpirationNotification","Version":"1.0","Attributes":{{"Before":"5","Unit":"Days"}},{"Type":"NoChangeNotification","Version":"1.0","Attributes":{{"After":"60","Unit":"Days"}}}]]"
```

3. Run the following command to verify the details of the parameter.

Linux

```bash
aws ssm describe-parameters \
  --parameter-filters "Key=Name,Values=parameter-name"
```

Windows

```bash
aws ssm describe-parameters ^ 
  --parameter-filters "Key=Name,Values=parameter-name"
```

**Important**

- Parameter Store retains policies for a parameter until you either overwrite the policies with new policies or remove the policies.
- To remove all policies from an existing parameter, edit the parameter and apply an empty policy of brackets and curly braces. For example:
Working with parameters

Linux

```bash
aws ssm put-parameter \
--name parameter-name \n--type parameter-type \n--value 'parameter-value' \n--policies "[{}]
```  

Windows

```bash
aws ssm put-parameter ^
--name parameter-name ^
--type parameter-type ^
--value 'parameter-value' ^
--policies "[{}]
```

- If you add a new policy to a parameter that already has policies, then Systems Manager overwrites the policies attached to the parameter. The existing policies are deleted. If you want to add a new policy to a parameter that already has one or more policies, then you must copy and paste the original policies, type the new policy, and then save your changes.

Add policies to an existing parameter by using the Tools for Windows PowerShell

Use the following procedure to add policies to an existing parameter by using Tools for Windows PowerShell.

To add policies to an existing parameter

1. Open AWS Tools for Windows PowerShell and run the following command to specify your credentials. You must either have administrator privileges in Amazon EC2, or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager prerequisites (p. 14).
   ```powershell
   Set-AWSCredentials –AccessKey access-key-name –SecretKey secret-key-name
   ```

2. Run the following command to set the Region for your PowerShell session. The example uses the US East (Ohio) Region (us-east-2).
   ```powershell
   Set-DefaultAWSRegion -Region us-east-2
   ```

3. Run the following command to add policies to an existing parameter.
   ```powershell
   Write-SSMParameter -Name "parameter-name" -Value "parameter-value" -Type "parameter-type" -Policies "["policies-enclosed-in-brackets-and-curly-braces"]" -Overwrite
   ```
   Here is an example that includes an expiration policy that deletes the parameter at midnight (GMT) on May 13, 2020. The example also includes a notification policy that generates a CloudWatch Events event five (5) days before the parameter is deleted. Last, it includes a NoChangeNotification policy if no changes are made to this parameter after 60 days. The example uses an obfuscated name (3l3vat3131) for a password and an AWS-managed customer master key (CMK).
   ```powershell
   Write-SSMParameter -Name "/Finance/Payroll/3l3vat3131" -Value "P@sSwW)rd" -Type "SecureString" -Policies "["Type":"Expiration","Version":"1.0","Attributes ":{"Timestamp":"2018-05-13T00:00:00Z"}],{"Type":"ExpirationNotification ","Version":"1.0","Attributes":{"Before":"5","Unit":"Days"}},{"Type":"NoChangeNotification","Attributes":null}
   ```
4. Run the following command to verify the details of the parameter.

```powershell
(Get-SSMParameterValue -Name "parameter-name-you-specified").Parameters
```

**Important**

- Parameter Store preserves policies on a parameter until you either overwrite the policies with new policies or remove the policies.
- To remove all policies from an existing parameter, edit the parameter and apply an empty policy of brackets and curly braces. For example:

```
Write-SSMParameter -Name "parameter-name" -Value "parameter-value" -Type "parameter-type" -Policies "{}"
```

- If you add a new policy to a parameter that already has policies, then Systems Manager overwrites the policies attached to the parameter. The existing policies are deleted. If you want to add a new policy to a parameter that already has one or more policies, then you must copy and paste the original policies, type the new policy, and then save your changes.

### About requirements and constraints for parameter names

Use the information in this topic to help you specify valid values for parameter names when you create a parameter.

This information supplements the details in the topic PutParameter in the AWS Systems Manager API Reference, which also provides information about the values AllowedPattern, Description, KeyId, Overwrite, Type, and Value.

The requirements and constraints for parameter names include the following:

- **Case sensitivity**: Parameter names are case sensitive.
- **Spaces**: Parameter names can't include spaces.
- **Valid characters**: Parameter names can consist of the following symbols and letters only: a-zA-Z0-9_.-/
- **Valid AMI format**: When you choose aws:ec2:image as the data type for a String parameter, the ID you enter must validate for the AMI ID format ami-12345abcdefEXAMPLE.
- **Fully qualified**: When you create or reference a parameter in a hierarchy, you must include a leading forward slash character (/). When you reference a parameter that is part of a hierarchy, you must specify the entire hierarchy path including the initial slash (/).
- **Fully qualified parameter names**: MyParameter1, /MyParameter2, /Dev/Production/East/Project-ABC/MyParameter
- **Not fully qualified parameter name**: MyParameter3/L1
- **Length**: The maximum length for a parameter name, including the full content of the parameter ARN, is 1011 characters. For example, the length of the following parameter is 65 characters, not 20 characters:

```
arn:aws:ssm:us-east-2:111122223333:parameter/ExampleParameterName
```

- **Prefixes**: A parameter name cannot be prefixed with "aws" or "ssm" (case-insensitive). For example, attempts to create parameters with the following names will fail with an exception:
• awsTestParameter
• SSM-testparameter
• /aws/testparam1

Note
When you specify a parameter in an SSM document, command, or script, you do include `ssm` as part of the syntax, as shown in the following examples.
• Valid: `{{ssm:parameter-name}}` and `{{ ssm:parameter-name }}`, such as `{{ssm:MyParameter}}`, and `{{ ssm:MyParameter }}`

- **Uniqueness**: A parameter name must be unique within an AWS Region. For example, Systems Manager treats the following as separate parameters, if they exist in the same Region:
  • /Test/TestParam1
  • /TestParam1

The following examples are also unique:
  • /Test/TestParam1/Logpath1
  • /Test/TestParam1

The following examples, however, if in the same Region, are not unique:
  • /TestParam1
  • TestParam1

- **Hierarchy depth**: If you specify a parameter hierarchy, the hierarchy can have a maximum depth of fifteen levels. You can define a parameter at any level of the hierarchy. Both of the following examples are structurally valid:
  • `/Level-1/L2/L3/L4/L5/L6/L7/L8/L9/L10/L11/L12/L13/L14/parameter-name`
  • `parameter-name`

Attempting to create the following parameter would fail with a `HierarchyLevelLimitExceeded` exception:
  • `/Level-1/L2/L3/L4/L5/L6/L7/L8/L9/L10/L11/L12/L13/L14/L15/L16/parameter-name`

**Important**
If a user has access to a path, then the user can access all levels of that path. For example, if a user has permission to access path `/a`, then the user can also access `/a/b`. Even if a user has explicitly been denied access in IAM for parameter `/a/b`, they can still call the `GetParametersByPath` API action recursively for `/a` and view `/a/b`.

**Creating Systems Manager parameters**

Use the information in the following topics to help you create Systems Manager parameters using the AWS Systems Manager console, the AWS CLI, or AWS Tools for Windows PowerShell.

**Topics**
- Create a Systems Manager parameter (console) (p. 249)
- Create a Systems Manager parameter (AWS CLI) (p. 251)
- Create a Systems Manager parameter (Tools for Windows PowerShell) (p. 255)

**Create a Systems Manager parameter (console)**

You can use the AWS Systems Manager console to create a Systems Manager parameter.

- **Note**
  Parameters are only available in the AWS Region where they were created.
To create a parameter

2. In the navigation pane, choose Parameter Store.
   
   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Parameter Store.
3. Choose Create parameter.
4. For Name, type a hierarchy and a parameter name. For example, type /Test/helloWorld.
5. In the Description box, type a description that identifies this parameter as a test parameter.
6. For Parameter tier choose either Standard or Advanced. For more information about advanced parameters, see Managing parameter tiers (p. 230).
7. For Type, choose String, StringList, or SecureString.
   
   - If you choose String, the Data type field appears. If you are creating a parameter to hold the resource ID for an Amazon Machine Image (AMI), select aws:ec2:image. Otherwise, leave the default text selected.
   
   - If you choose SecureString, the KMS Key ID field appears. If you don't provide a KMS customer master key (CMK) ID, a CMK ARN, an alias name, or an alias ARN, then the system uses alias/aws/ssm, which is the AWS managed CMK for Systems Manager. If you don't want to use this key, then you can use a customer managed CMK. For more information about SecureString parameters, see SecureString parameters (p. 206). For more information about AWS managed and customer managed CMKs, see AWS Key Management Service Concepts in the AWS Key Management Service Developer Guide. For more information about Parameter Store and KMS encryption, see How AWS Systems Manager Parameter Store Uses AWS KMS.
   
   - When creating a SecureString parameter in the console by using the key-id parameter with either a customer managed CMK alias name or an alias ARN, you must specify the prefix alias/ before the alias. Here is an ARN example:

     ```
     arn:aws:kms:us-east-2:123456789012:alias/MyAliasName
     ```

   Here is an alias name example:

   ```
   alias/MyAliasName
   ```

8. In the Value box, type a value. For example, type This is my first parameter or ami-0dbf5ea29aEXAMPLE. If you chose SecureString, the value is masked as you type.
9. (Optional) In the Tags area, apply one or more tag key-value pairs to the parameter.

   Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a Systems Manager parameter to identify the type of resource to which it applies, the environment, or the type of configuration data referenced by the parameter. In this case, you could specify the following key-value pairs:

   ```
   • Key=Resource,Value=S3bucket
   • Key=OS,Value=Windows
   • Key=ParameterType,Value=LicenseKey
   ```
10. Choose Create parameter.
11. In the parameters list, choose the name of the parameter you just created. Verify the details on the Overview tab. If you created a SecureString parameter, choose Show to view the unencrypted value.
Note
You can’t change an advanced parameter to a standard parameter. If you no longer need an advanced parameter, or if you no longer want to incur charges for an advanced parameter, you must delete it and recreate it as a new standard parameter.

Create a Systems Manager parameter (AWS CLI)

You can use the AWS CLI to create String, StringList, and SecureString parameter types.

For more information about using the AWS CLI to create parameters, see Walkthrough: Create and update a String parameter (AWS CLI) (p. 267).

Note
Parameters are only available in the AWS Region where they were created.

Topics
• Create a String Parameter (AWS CLI) (p. 251)
• Create a StringList parameter (AWS CLI) (p. 252)
• Create a SecureString parameter (AWS CLI) (p. 253)

Create a String Parameter (AWS CLI)

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Run the following command to create a parameter that contains a plain text value.
   
   Linux
   ```bash
   aws ssm put-parameter \
   --name "parameter-name" \
   --value "parameter-value" \
   --type String \
   --tags "Key=tag-key,Value=tag-value"
   ```

   Windows
   ```bash
   aws ssm put-parameter ^
   --name "parameter-name" ^
   --value "parameter-value" ^
   --type String ^
   --tags "Key=tag-key,Value=tag-value"
   ```

   or

   Run the following command to create a parameter that contains an Amazon Machine Image (AMI) ID as the parameter value.
   
   Linux
   ```bash
   aws ssm put-parameter \
   --name "parameter-name" \
   --value "an-AMI-id" \
   --type String \
   --data-type "aws:ec2:image" \
   --tags "Key=tag-key,Value=tag-value"
   ```
Windows

aws ssm put-parameter ^
   --name "parameter-name" ^
   --value "an-AMI-id" ^
   --type String ^
   --data-type "aws:ec2:image" ^
   --tags "Key=tag-key,Value=tag-value"

The --data-type option must be specified only if you are creating a parameter that contains an AMI ID. For all other parameters, the default data type is text. For more information, see Native parameter support for Amazon Machine Image IDs (p. 210).

Important
If successful, the command returns the version number of the parameter. Exception: If you have specified aws:ec2:image as the data type, a new version number in the response does not mean that the parameter value has been validated yet. For more information, see Native parameter support for Amazon Machine Image IDs (p. 210).

The following example adds two key-value pair tags to a parameter.

Linux

aws ssm put-parameter \ 
   --name parameter-name \ 
   --value "parameter-value" \ 
   --type "String" \ 
   --tags "[\{"Key\":\"Region\",\"Value\":\"East\"\},\{"Key\":\"Environment\",\n   \"Value\":\"Production\"\}]"

Windows

aws ssm put-parameter ^
   --name parameter-name ^
   --value "parameter-value" ^
   --type "String" ^
   --tags "[\{"Key\":\"Region1\",\"Value\":\"East1\"\},\{"Key\":\"Environment1\",\n   \"Value\":\"Production1\"\}]"

3. Run the following command to verify the details of the parameter.

aws ssm get-parameters --name "parameter-name"

Create a StringList parameter (AWS CLI)

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create a parameter.

   Linux

   aws ssm put-parameter \ 
      --name "parameter-name" \ 
      --value "a-comma-separated-list-of-values" \ 
      --type StringList \ 

--tags "Key=tag-key,Value=tag-value"

Windows

aws ssm put-parameter
--name "parameter-name" ^
--value "a-comma-separated-list-of-values" ^
--type StringList ^
--tags "Key=tag-key,Value=tag-value"

**Note**
If successful, the command returns the version number of the parameter.

This example adds two key-value pair tags to a parameter. (Depending on the operating system type on your local machine, run one of the following commands. The version to run from a local Windows machine includes the escape characters ("\") that you need to run the command from your command line tool.)

Here is a **StringList** example that uses a parameter hierarchy.

**Linux**

aws ssm put-parameter \
--name /IAD/ERP/Oracle/addUsers \
--value "Milana,Mariana,Mark,Miguel" \
--type StringList

**Windows**

aws ssm put-parameter ^
--name /IAD/ERP/Oracle/addUsers ^
--value "Milana,Mariana,Mark,Miguel" ^
--type StringList

**Note**
Items in a **StringList** must be separated by a comma (,). You can't use other punctuation or special character to escape items in the list. If you have a parameter value that requires a comma, then use the **String** type.

3. Run the **get-parameters** command to verify the details of the parameter. For example:

```bash
aws ssm get-parameters --name "/IAD/ERP/Oracle/addUsers"
```

**Create a SecureString parameter (AWS CLI)**

Before you create a **SecureString** parameter, read about the requirements for this type of parameter. For more information, see **SecureString parameters** (p. 206).

1. Install and configure the AWS CLI, if you have not already.
   For information, see **Install or upgrade AWS command line tools** (p. 57).
2. Run the following command to create a parameter.
Linux

```bash
aws ssm put-parameter \
--name "parameter-name" \
--value "parameter-value" \
--type SecureString \
--key-id "a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN" \
--tags "Key=tag-key,Value=tag-value"
```

Windows

```bash
aws ssm put-parameter ^ 
--name "parameter-name" ^ 
--value "parameter-value" ^ 
--type SecureString ^ 
--key-id "a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN" ^ 
--tags "Key=tag-key,Value=tag-value"
```

**Note**

To use the AWS Key Management Service (KMS) customer master key (CMK) assigned to your account, remove the `key-id` parameter from the command. For more information about CMKs, see [AWS Key Management Service Concepts](https://docs.aws.amazon.com/kms/latest/developerguide/) in the [AWS Key Management Service Developer Guide](https://docs.aws.amazon.com/kms/latest/developerguide/).

The following example uses an obfuscated name (`3l3vat3131`) for a password parameter and a CMK.

Linux

```bash
aws ssm put-parameter \
--name /Finance/Payroll/3l3vat3131 \
--value "P@sSwW)rd" \
--type SecureString \
--key-id arn:aws:kms:us-east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d5e
```

Windows

```bash
aws ssm put-parameter ^ 
--name /Finance/Payroll/3l3vat3131 ^ 
--value "P@sSwW)rd" ^ 
--type SecureString ^ 
--key-id arn:aws:kms:us-east-2:123456789012:key/1a2b3c4d-1a2b-1a2b-1a2b3c4d5e
```

3. Run the following command to verify the details of the parameter.

Linux

```bash
aws ssm get-parameters \
--name "the-parameter-name-you-specified" \
--with-decryption
```

Windows

```bash
aws ssm get-parameters ^
```
Working with parameters

```
--name "the-parameter-name-you-specified" ^
--with-decryption
```

**Note**

If you don’t specify the `--with-decryption` parameter, or if you specify the `--no-with-decryption` parameter, the command returns an encrypted GUID.

---

### Create a Systems Manager parameter (Tools for Windows PowerShell)

You can use Tools for Windows PowerShell to create String, StringList, and SecureString parameter types.

**Note**

Parameters are only available in the AWS Region where they were created.

**Topics**

- Create a String parameter (Tools for Windows PowerShell) (p. 255)
- Create a StringList parameter (Tools for Windows PowerShell) (p. 256)
- Create a SecureString parameter (Tools for Windows PowerShell) (p. 256)

#### Create a String parameter (Tools for Windows PowerShell)

1. Install and configure the AWS Tools for PowerShell, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create a parameter that contains a plain text value.

   ```powershell
   Write-SSMParameter
   -Name "parameter-name"
   -Value "parameter-value"
   -Type "String"
   -or-
   Write-SSMParameter
   -Name "parameter-name"
   -Value "an-AMI-id"
   -Type "String"
   -DataType "aws:ec2:image"
   -Tags "Key=tag-key,Value=tag-value"
   ```

   The `--DataType` option must be specified only if you are creating a parameter that contains an AMI ID. For all other parameters, the default data type is `text`. For more information, see Native parameter support for Amazon Machine Image IDs (p. 210).

Here is a String example that uses a parameter hierarchy.

```powershell
Write-SSMParameter
-Name "/IAD/Web/SQL/IPaddress"
-Value "99.99.99.999"
-Type "String"
```
Tags "Key=Region,Value=IAD"

3. Run the following command to verify the details of the parameter.

```
(Get-SSMParameterValue -Name "the-parameter-name-you-specified").Parameters
```

Create a **StringList** parameter (Tools for Windows PowerShell)

1. Install and configure the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create a StringList parameter.

```
Write-SSMParameter `-Name "parameter-name" `-Value "a-comma-separated-list-of-values" `-Type "StringList" `-Tags "Key=tag-key,Value=tag-value"
```

If successful, the command returns the version number of the parameter.

**Note**

Items in a **StringList** must be separated by a comma (,). You can't use other punctuation or special character to escape items in the list. If you have a parameter value that requires a comma, then use the **String** type.

3. Run the following command to verify the details of the parameter.

```
(Get-SSMParameterValue -Name "the-parameter-name-you-specified").Parameters
```

Create a **SecureString** parameter (Tools for Windows PowerShell)

Before you create a **SecureString** parameter, read about the requirements for this type of parameter. For more information, see **SecureString** parameters (p. 206).

1. Install and configure the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create a SecureString parameter.

```
Write-SSMParameter `-Name "parameter-name" `-Value "parameter-value" `-Type "SecureString" `-KeyId "a KMS CMK ID, a KMS CMK ARN, an alias name, or an alias ARN" `-Tags "Key=tag-key,Value=tag-value"
```

If successful, the command returns the version number of the parameter.

**Note**

To use the AWS-managed customer master key (CMK) assigned to your account, remove the -KeyId parameter from the command.

Here is an example that uses an obfuscated name (3l3vat3131) for a password parameter and an AWS-managed customer master key (CMK).
Searching for Systems Manager parameters

When you have a large number of parameters in your account, it can be difficult to find information about just one or a few parameters at a time. In this case, you can use filter tools to search for the ones you need information about, according to search criteria you specify. You can use the Systems Manager console, the AWS CLI, the AWS Tools for PowerShell, or the DescribeParameters API to search for parameters.

For information about using the Systems Manager console and AWS CLI to view information about parameters in your account by using search, or filter options, see the following topics.

Topics
- Search for a parameter (console) (p. 257)
- Search for a parameter (AWS CLI) (p. 257)

Search for a parameter (console)

2. In the left navigation, choose Parameter Store.
3. Click in the search box and choose how you want to search. For example, Type or Name.
4. Provide information for the search type you selected. For example:
   - If you are searching by Type, choose from String, StringList, or SecureString.
   - If you are searching by Name, choose contains, equals, or begins-with, and then enter all or part of a parameter name.

   **Note**
   In the console, the default search type for Name is contains.

The list of parameters is updated with the results of your search.

Search for a parameter (AWS CLI)

Use the describe-parameters command to view information about one or more parameters in the AWS CLI.

The following examples demonstrate various options you can use to view information about the parameters in your AWS account. For more information about these options, see describe-parameters in the AWS Command Line Interface User Guide.

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).
2. Replace the sample values in the following commands with values reflecting parameters that have been created in your account.

Linux

```bash
aws ssm describe-parameters
   --parameter-filters "Key=Name,Values=MyParameterName"
```

Windows

```bash
aws ssm describe-parameters
   --parameter-filters "Key=Name,Values=MyParameterName"
```

**Note**

For `describe-parameters`, the default search type for `Name` is `Equals`. In your parameter filters, specifying "Key=Name,Values=MyParameterName" is the same as specifying "Key=Name,Option=Equals,Values=MyParameterName".

```bash
aws ssm describe-parameters --parameter-filters
   "Key=Name,Option=Contains,Values=Product"
```

```bash
aws ssm describe-parameters --parameter-filters "Key=Type,Values=String"
```

```bash
aws ssm describe-parameters --parameter-filters "Key=Path,Values=/Production/West"
```

```bash
aws ssm describe-parameters --parameter-filters "Key=Tier,Values=Standard"
```

```bash
aws ssm describe-parameters --parameter-filters "Key=tag:tag-key,Values=tag-value"
```

```bash
aws ssm describe-parameters --parameter-filters "Key=KeyId,Values=key-id"
```

**Note**

In the last example, `key-id` represents the ID of a AWS Key Management Service (AWS KMS) key used to encrypt a SecureString parameter created in your account. Alternatively, you can enter `alias/aws/ssm` to use the default AWS KMS key for your account. For more information, see SecureString parameters (p. 206).

If successful, the command returns output similar to the following.

```json
{
   "Parameters": [
      {
         "Name": "/Production/West/Manager",
         "Type": "String",
         "LastModifiedDate": 1573438580.703,
         "LastModifiedUser": "arn:aws:iam::111122223333:user/Mateo.Jackson",
         "Version": 1,
         "Tier": "Standard",
         "Policies": []
      },
      {
         "Name": "/Production/West/TeamLead",
         "Type": "String",
```
Working with parameter versions

Each time you edit the value of a parameter, Parameter Store creates a new version of the parameter and retains the previous versions. When you initially create a parameter, Parameter Store assigns version 1 to that parameter. When you change the value of the parameter, Parameter Store automatically iterates the version number by one. You can view the details, including the values, of all versions in a parameter's history.

You can also specify the version of a parameter to use in API commands and SSM documents; for example: `ssm:MyParameter:3`. You can specify a parameter name and a specific version number in API calls and SSM documents. If you don't specify a version number, the system automatically uses the latest version.

You can use parameter versions to see how many times a parameter changed over a period of time. Parameter versions also provide a layer of protection if a parameter value is accidentally changed.

You can create a maximum of 100 versions of a parameter.

The following procedures show you how to edit a parameter and then verify that a new version was created.

**Topics**

- Create a new version of a parameter (console) (p. 259)
- Reference a parameter version (p. 260)

**Create a new version of a parameter (console)**

You can use the AWS Systems Manager console to create a new version of a parameter and view the version history of a parameter.

**To create a new version of a parameter**

2. In the navigation pane, choose **Parameter Store**.
   
   -or-
   
   If the AWS Systems Manager home page opens first, choose the menu icon (Ξ) to open the navigation pane, and then choose **Parameter Store**.
3. Choose the name of a parameter that you created earlier. For information about creating a new parameter, see Creating Systems Manager parameters (p. 249).
4. Choose Edit.
5. In the Value box, type a new value, and then choose Save changes.
6. Choose the name of the parameter you just updated. On the Overview tab, verify that the version number incremented by 1, and verify the new value.
7. To view the history of all versions of a parameter, choose the History tab.

Reference a parameter version

You can reference specific parameter versions in commands, API calls, and SSM documents by using the following format: ssm: parameter-name:version-number.

In the following example, version 3 of the parameter golden-ami is used in the EC2 run-instances command.

```bash
aws ec2 run-instances \
  --image-id resolve:ssm:/golden-ami:3 \
  --count 1 \
  --instance-type t2.micro \
  --key-name my-key-pair \
  --security-groups my-security-group
```

**Note**

Currently, using resolve and a parameter value works only with the --instance-id option and a parameter that contains an Amazon Machine Image (AMI) as its value. For more information, see Native parameter support for Amazon Machine Image IDs (p. 210).

Here is an example for specifying version 2 of a parameter named MyRunCommandParameter in an SSM document.

```json
{
  "schemaVersion": "2.2",
  "description": "Run a shell script or specify the commands to run.",
  "parameters": {
    "commands": {
      "type": "String",
      "description": "(Required) Specify a shell script or a command to run.",
      "displayType": "textarea",
      "default": "{{ssm:MyRunCommandParameter:2}}"
    }
  },
  "mainSteps": [
    {
      "action": "aws:runShellScript",
      "name": "RunScript",
      "inputs": {
        "runCommand": [
          "{{commands}}"
        ]
      }
    }
  ]
}
```

Labeling parameters

A parameter label is a user-defined alias to help you manage different versions of a parameter. When you modify a parameter, Systems Manager automatically saves a new version and increments the version.
number by one. A label can help you remember the purpose of a parameter version when there are multiple versions.

For example, let’s say you have a parameter called /MyApp/DB/ConnectionString. The value of the parameter is a connection string to a MySQL server in a local database in a test environment. After you finish updating the application, you want the parameter to use a connection string for a production database. You change the value of /MyApp/DB/ConnectionString. Systems Manager automatically creates version two with the new connection string. To help you remember the purpose of each version, you attach a label to each parameter. For version one, you attach the label Test and for version two you attach the label Production.

You can move labels from one version of a parameter to another version. For example, if you create version three of the /MyApp/DB/ConnectionString parameter with a connection string for a new production database, then you can move the Production label from parameter two to parameter three.

Parameter labels are a lightweight alternative to parameter tags. Your organization might have strict guidelines for tags that must be applied to different AWS resources. In contrast, a label is simply a text association for a specific version of a parameter.

Similar to tags, you can query parameters by using labels. You can view a list of specific parameter versions that all use the same label if you query your parameter set by using the GetParametersByPath API action, as described later in this section.

**Label Requirements and Restrictions**

Parameter labels have the following requirements and restrictions:

- A version of a parameter can have a maximum of 10 labels.
- You can’t attach the same label to different versions of the same parameter. For example, if version 1 has the label Production, then you can’t attach Production to version 2.
- You can move a label from one version of a parameter to another.
- You can’t create a label when you create a new parameter. You must attach a label to a specific version of a parameter.
- You can’t delete a parameter label. If you no longer want to use a parameter label, then you must move it to a different version of a parameter.
- A label can have a maximum of 100 characters.
- Labels can contain letters (case sensitive), numbers, periods (.), hyphens (-), or underscores (_).
- Labels can’t begin with a number, "aws," or "ssm" (not case sensitive). If a label fails to meet these requirements, then the label is not attached to the parameter version and the system displays it in the list of InvalidLabels.

**Topics**

- Working with parameter labels (console) (p. 261)
- Working with parameter labels (AWS CLI) (p. 263)

**Working with parameter labels (console)**

This section describes how to perform the following tasks by using the AWS Systems Manager console.

- Create a new parameter label (p. 262)
- View labels attached to a parameter (p. 262)
- Move a parameter label (p. 262)
Create a new parameter label

The following procedure describes how to attach a label to a specific version of an existing parameter by using the Systems Manager console. You can't attach a label when you create a parameter.

To attach a label to a parameter version by using the console

2. In the navigation pane, choose Parameter Store.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (☰) to open the navigation pane, and then choose Parameter Store.
3. Choose the name of a parameter to open the details page for that parameter.
4. Choose the History tab.
5. Choose the parameter version for which you want to attach a label.
6. Choose Attach labels.
7. Choose Add another label.
8. In the text box, enter the label. To add more labels, choose Add another label. You can attach a maximum of ten labels.
9. When you are finished attaching labels, choose Confirm.

View labels attached to a parameter

A parameter version can have a maximum of ten labels. The following procedure describes how to view all labels attached to a parameter version by using the Systems Manager console.

To view labels attached to a parameter version by using the console

2. In the navigation pane, choose Parameter Store.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (☰) to open the navigation pane, and then choose Parameter Store.
3. Choose the name of a parameter to open the details page for that parameter.
4. Choose the History tab.
5. Locate the parameter version for which you want to view all attached labels. The Labels column shows all labels attached to the parameter version.

Move a parameter label

You can't delete a parameter label after you create it. You can, however, move a label between versions of a parameter. The following procedure describes how to move a parameter label to a different version of the same parameter by using the Systems Manager console.

To move a label to a different parameter version by using the console

2. In the navigation pane, choose Parameter Store.
-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Parameter Store.

3. Choose the name of a parameter to open the details page for that parameter.
4. Choose the History tab.
5. Choose the parameter version for which you want to move the label.
6. Choose Attach labels.
7. Choose Add another label.
8. In the text box, enter the label. The console notifies you of the label move.
9. When you are finished, choose Confirm.

Working with parameter labels (AWS CLI)

This section describes how to perform the following tasks by using the AWS CLI.

- Create a new parameter label (p. 263)
- View labels for a parameter (p. 264)
- View a list of parameters that are assigned a label (p. 265)
- Move a parameter label (p. 266)

Create a new parameter label

The following procedure describes how to attach a label to a specific version of an existing parameter by using the AWS CLI. You can't attach a label when you create a parameter.

To create a new parameter label

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Run the following command to view a list of parameters for which you have permission to attach a label.

   \[\text{Note} \]
   Parameters are only available in the Region where they were created. If you don't see a parameter for which you want to attach a label, then verify your Region.

   \[\text{aws ssm describe-parameters}\]

   Note the name of a parameter for which you want to attach a label.
3. Run the following command to view all versions of the parameter.

   \[\text{aws ssm get-parameter-history --name "parameter-name"}\]

   Note the parameter version for which you want to attach a label.
4. Run the following command to retrieve information about a parameter by version number.

   \[\text{aws ssm get-parameters --names "parameter-name:version-number"}\]

   Here is an example.
5. Run one of the following commands to attach a label to a version of a parameter. If you attach multiple labels, then you must separate label names with a space.

**Attach a label to the latest version of a parameter**

```
aws ssm label-parameter-version --name parameter-name --labels label-name
```

**Attach a label to a specific version of a parameter**

```
aws ssm label-parameter-version --name parameter-name --parameter-version version-number --labels label-name
```

Here are some examples:

```
aws ssm label-parameter-version --name /config/endpoint --labels production east-region finance

aws ssm label-parameter-version --name /config/endpoint --parameter-version 3 --labels MySQL-test
```

**Note**

If the output shows the label you created in the InvalidLabels list, then the label does not meet the requirements described earlier in this topic. Review the requirements and try again. If the InvalidLabels list is empty, then your label was successfully applied to the version of the parameter.

6. You can view the details of the parameter by using either a version number or a label name. Run the following command and specify the label you created in the previous step.

```
aws ssm get-parameter --name parameter-name:label-name --with-decryption
```

The command returns information like the following:

```
{
  "Parameter": {
    "Version": version-number,
    "Type": "parameter-type",
    "Name": "parameter-name",
    "Value": "parameter-value",
    "Selector": ":label-name"
  }
}
```

**Note**

Selector in the output is either the version number or the label that you specified in the Name input field.

**View labels for a parameter**

You can use the GetParameterHistory API action to view the full history and all labels attached to a specified parameter. Or, you can use the GetParametersByPath API action to view a list of all parameters that are assigned a specific label.
To view labels for a parameter by using the GetParameterHistory API action

1. Run the following command to view a list of parameters for which you can view labels.

   **Note**
   Parameters are only available in the Region where they were created. If you don’t see a parameter for which you want to move a label, then verify your Region.

   ```
   aws ssm describe-parameters
   ```

   Note the name of a parameter for which you want to move a label.

2. Run the following command to view all versions of the parameter.

   ```
   aws ssm get-parameter-history --name parameter-name --with-decryption
   ```

   The system returns information like the following:

   ```
   {
     "Parameters": [
       {
         "Name": "/Config/endpoint",
         "LastModifiedDate": 1528932105.382,
         "Labels": [
           "Deprecated"
         ],
         "Value": "MyTestService-June-Release.example.com",
         "Version": 1,
         "LastModifiedUser": "arn:aws:iam::123456789012:user/test",
         "Type": "String"
       },
       {
         "Name": "/Config/endpoint",
         "LastModifiedDate": 1528932111.222,
         "Labels": [
           "Current"
         ],
         "Value": "MyTestService-July-Release.example.com",
         "Version": 2,
         "LastModifiedUser": "arn:aws:iam::123456789012:user/test",
         "Type": "String"
       }
     ]
   }
   ```

View a list of parameters that are assigned a label

You can use the GetParametersByPath API action to view a list of all parameters in a path that are assigned a specific label.

Run the following command to view a list of parameters in a path that are assigned a specific label.

```
aws ssm get-parameters-by-path --path parameter-path --parameter-filters
Key=Label,Values=label-name,Option=Equals --max-results a-number --with-decryption --recursive
```

The system returns information like the following. For this example, the user searched under the /Config path:
Move a parameter label

You can't delete a parameter label after you create it. You can, however, move a label between versions of a parameter. The following procedure describes how to move a parameter label to a different version of the same parameter.

To move a parameter label

1. Run the following command to view all versions of the parameter.

```
aws ssm get-parameter-history --name "parameter-name"
```

Note the parameter version for which you want to attach a label.

2. Run the following command to assign an existing label to a different version of a parameter.

```
aws ssm label-parameter-version --name parameter-name --parameter-version version-number --labels name-of-existing-label
```

Note
If you want to move an existing label to the latest version of a parameter, then remove --parameter-version from the command.

Parameter Store walkthroughs

The walkthroughs in this section show you how to create, store, and run parameters with Parameter Store in a test environment. These walkthroughs show you how to use Parameter Store with other Systems Manager capabilities. You can also use Parameter Store with other AWS services. For more information, see Using SecureString parameters with other AWS services (p. 210).

Contents

- Walkthrough: Create and test a String parameter (console) (p. 267)
- Walkthrough: Create and update a String parameter (AWS CLI) (p. 267)
- Walkthrough: Create and update a SecureString parameter (AWS CLI) (p. 269)
Walkthrough: Create and test a String parameter (console)

The following procedure walks you through the process of creating a String parameter in Parameter Store and then running a command that uses this parameter.

To create a String parameter using Parameter Store

2. In the navigation pane, choose Parameter Store.
3. Choose Create parameter.
4. In the Name box, enter a hierarchy and a name. For example, enter /Test/helloWorld.
   For more information about parameter hierarchies, see Organizing parameters into hierarchies (p. 240).
5. In the Description field, enter a description that identifies this parameter as a test parameter.
6. For Type, choose String.
7. For Data type, leave the default selection text.
8. In the Value field, enter a string. For example, enter This is my first parameter.
9. Choose Create parameter.
10. In the navigation pane, choose Run Command.
11. Choose Run command.
12. In the Command document list, choose AWS-RunPowerShellScript (Windows) or AWS-RunShellScript (Linux).
13. For Command parameters, enter echo {{ssm:parameter-name}}. For example: echo {{ssm:/Test/helloWorld}}.
14. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.
   Note
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.
15. Choose Run.
16. In the Command ID page, in the Targets and outputs area, select the button next to the ID of an instance where you ran the command, and then choose View output. Verify that the output of the command is the value you provided for the parameter, such as This is my first parameter.

Walkthrough: Create and update a String parameter (AWS CLI)

The following procedure walks you through the process of creating and storing a parameter of the type String using the AWS CLI.

To create and use a String parameter in a command (AWS CLI)

1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Run the following command to create a String-type parameter. The --name option supports hierarchies. For information about hierarchies, see Organizing parameters into hierarchies (p. 240).
Here is an example that specifies an Amazon Machine Image (AMI) as the data type:

```bash
aws ssm put-parameter --name "golden-ami" --type String --data-type "aws:ec2:image" --value ami-0dbf5ea29aEXAMPLE
```

The option `--data-type` applies to String-type parameters only and is required only if you are specifying `aws:ec2:image` as the data type. You do not need to explicitly specify a data type if the String data type is text.

Here is an example that uses a parameter hierarchy in the name. For more information about parameter hierarchies, see Organizing parameters into hierarchies (p. 240).

```bash
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "My 1st parameter" --type String
```

The command returns the version number of the parameter.

3. Run the following command to view the parameter metadata.

```bash
aws ssm describe-parameters --filters "Key=Name,Values="/Test/IAD/helloWorld"
```

**Note**

*Name* must be capitalized.

The system returns information like the following.

```json
{
  "Parameters": [
    {
      "Name": "helloworld",
      "Type": "String",
      "LastModifiedUser": "arn:aws:iam::123456789012:user/user-name",
      "LastModifiedDate": 1494529763.156,
      "Version": 1,
      "Tier": "Standard",
      "Policies": []
    }
  ]
}
```

4. Run the following command to change the parameter value.

```bash
aws ssm put-parameter --name "/Test/IAD/helloWorld" --value "My updated 1st parameter" --type String --overwrite
```

The command returns the version number of the parameter.

5. Run the following command to view the latest parameter value.

```bash
aws ssm get-parameters --names "/Test/IAD/helloWorld"
```

The system returns information like the following.

```json
{
}
```
"InvalidParameters": [],
"Parameters": [
    {
        "Name": "/Test/IAD/helloWorld",
        "Type": "String",
        "Value": "My updated parameter value",
        "Version": 2,
        "LastModifiedDate": "2020-02-25T15:55:33.677000-08:00",
        "ARN": "arn:aws:ssm:us-east-2:123456789012:parameter/Test/IAD/helloWorld"
    }
]

6. Run the following command to view the parameter value history.

    aws ssm get-parameter-history --name "/Test/IAD/helloWorld"

7. Run the following command to use this parameter in a command.

    aws ssm send-command --document-name "AWS-RunShellScript" --parameters '{"commands": [
        "echo {{ssm:/Test/IAD/helloWorld}}"]}' --targets "Key=instanceids,Values=instance-ids"

Walkthrough: Create and update a SecureString parameter (AWS CLI)

Use the following procedure to create a SecureString parameter. For more information about SecureString parameters, see SecureString parameters (p. 206).

To create a SecureString parameter using the AWS CLI

1. Run one of the following commands to create a parameter that uses the SecureString datatype.

    **Linux**

    **Create a SecureString parameter that uses a customer managed customer master key (CMK)**

    ```
    aws ssm put-parameter \
        --name "parameter-name" \
        --value "a-parameter-value, for example P@ssW%rd#1" \
        --type "SecureString"
    ```

    **Create a SecureString parameter that uses a custom AWS KMS key**

    ```
    aws ssm put-parameter \
        --name "parameter-name" \
        --value "a-parameter-value, for example P@ssW%rd#1" \
        --type "SecureString" \
        --key-id "your-AWS-user-account-ID/the-custom-AWS KMS-key"
    ```

    **Windows**

    **Create a SecureString parameter that uses a customer managed customer master key (CMK)**

    ```
    aws ssm put-parameter ^
Create a SecureString parameter that uses a custom AWS KMS key

aws ssm put-parameter
  --name "parameter-name"
  --value "a-parameter-value, for example P@ssW%rd#1"
  --type "SecureString"
  --key-id "your-AWS-user-account-ID/the-custom-AWS KMS-key"

Here is an example that uses a customer managed CMK.

Linux

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-1a2EXAMPLE"

Windows

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-1a2EXAMPLE"

2. Run the following command to view the parameter metadata.

Linux

aws ssm describe-parameters
  --filters "Key=Name,Values=the-name-that-you-specified"

Windows

aws ssm describe-parameters
  --filters "Key=Name,Values=the-name-that-you-specified"

3. Run the following command to change the parameter value if you are not using a customer managed customer master key (CMK).

Linux

aws ssm put-parameter
  --name "the-name-that-you-specified"
  --value "a-new-parameter-value"
  --type "SecureString"
  --overwrite
Windows

aws ssm put-parameter
  --name "the-name-that-you-specified" ^
  --value "a-new-parameter-value" ^
  --type "SecureString" ^
  --overwrite

-or-

Run one the following commands to change the parameter value if you are using a customer managed customer master key (CMK).

Linux

aws ssm put-parameter \n  --name "the-name-that-you-specified" \n  --value "a-new-parameter-value" \n  --type "SecureString" \n  --key-id "the-CMK-ID" \n  --overwrite

aws ssm put-parameter \n  --name "the-name-that-you-specified" \n  --value "a-new-parameter-value" \n  --type "SecureString" \n  --key-id "your-AWS-user-account-alias/the-CMK-ID" \n  --overwrite

Windows

aws ssm put-parameter ^
  --name "the-name-that-you-specified" ^
  --value "a-new-parameter-value" ^
  --type "SecureString" ^
  --key-id "the-CMK-ID" ^
  --overwrite

aws ssm put-parameter ^
  --name "the-name-that-you-specified" ^
  --value "a-new-parameter-value" ^
  --type "SecureString" ^
  --key-id "your-AWS-user-account-alias/the-CMK-ID" ^
  --overwrite

4. Run the following command to view the latest parameter value.

Linux

aws ssm get-parameters \n  --name "the-name-that-you-specified" \n  --with-decryption

Windows

aws ssm get-parameters ^
5. Run the following command to view the parameter value history.

Linux

```bash
aws ssm get-parameter-history --name "the-name-that-you-specified"
```

Windows

```bash
aws ssm get-parameter-history --name "the-name-that-you-specified"
```

**Important**

Only the value of a SecureString parameter is encrypted. Parameter names, descriptions, and other properties are not encrypted.

## Walkthrough: Create a SecureString parameter and join an instance to a Domain (PowerShell)

This walkthrough shows how to join a Windows Server instance to a domain using Systems Manager SecureString parameters and Run Command. The walkthrough uses typical domain parameters, such as the domain name and a domain user name. These values are passed as unencrypted string values. The domain password is encrypted using an AWS-managed customer master key (CMK) and passed as an encrypted string.

### Prerequisites

This walkthrough assumes that you already specified your domain name and DNS server IP address in the DHCP option set that is associated with your Amazon VPC. For information, see [Working with DHCP Options Sets](https://docs.aws.amazon.com/vpc/latest/userguide/dhcopp.html) in the Amazon VPC User Guide.

### To create a SecureString parameter and join an instance to a domain

1. Enter parameters into the system using AWS Tools for Windows PowerShell.

   ```powershell
   Write-SSMParameter -Name "domainName" -Value "DOMAIN-NAME" -Type String
   Write-SSMParameter -Name "domainJoinUserName" -Value "DOMAIN\USERNAME" -Type String
   Write-SSMParameter -Name "domainJoinPassword" -Value "PASSWORD" -Type SecureString
   ```

   **Important**
   Only the value of a SecureString parameter is encrypted. Parameter names, descriptions, and other properties are not encrypted.

2. Attach the following IAM policies to the IAM role permissions for your instance:

   - **AmazonSSMManagedInstanceCore** – Required. This AWS managed policy enables a managed instance to use Systems Manager service core functionality.
   - **AmazonSSMDirectoryServiceAccess** – Required. This AWS managed policy allows SSM Agent to access AWS Directory Service on your behalf for requests to join the domain by the managed instance.
   - **A custom policy for S3 bucket access** – Required. SSM Agent, which is on your instance and performs Systems Manager tasks, requires access to specific Amazon-owned S3 buckets. In the
custom S3 bucket policy that you create, you also provide access to S3 buckets of your own that are necessary for Systems Manager operations.

Examples: You can write output for Run Command commands or Session Manager sessions to an S3 bucket, and then use this output later for auditing or troubleshooting. You store access scripts or custom patch baseline lists in an S3 bucket, and then reference the script or list when you run a command, or when a patch baseline is applied.

For information about creating a custom policy for Amazon S3 bucket access, see Create a custom S3 bucket policy for an instance profile (p. 32)

**Note**

Saving output log data in an S3 bucket is optional, but we recommend setting it up at the beginning of your Systems Manager configuration process if you have decided to use it.

For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide.

- **CloudWatchAgentServerPolicy** – Optional. This AWS managed policy allows you to run the CloudWatch agent on managed instances. This policy makes it possible to read information on an instance and write it to Amazon CloudWatch. Your instance profile needs this policy only if you will use CloudWatch features, such as CloudWatch Events or CloudWatch Logs.

  **Note**

  Using CloudWatch features is optional, but we recommend setting them up at the beginning of your Systems Manager configuration process if you have decided to use them. For more information, see the Amazon CloudWatch Events User Guide and the Amazon CloudWatch Logs User Guide.

3. Edit the IAM role attached to the instance and add the following policy. This policy gives the instance permissions to call the `kms:Decrypt` API.

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect":"Allow",
            "Action":
            ["kms:Decrypt"],
            "Resource":
            ["arn:aws:kms:region:account-id:key/kms-key-id"]
        }
    ]
}
```

4. Copy and paste the following json sample into a simple text editor and save the file as `JoinInstanceToDomain.json` in the following location: `c:\temp\JoinInstanceToDomain.json`.

```
{
    "schemaVersion": "2.2",
    "description": "Run a PowerShell script to securely domain-join a Windows Server instance",
    "mainSteps": [
        {
            "action": "aws:runPowerShellScript",
            "name": "runPowerShellWithSecureString",
            "precondition": {
                "StringEquals": {
                    "platformType": "Windows"
                }
            }
        }
    ]
}
```

273
### AWS Tools for Windows PowerShell

5. Run the following command in AWS Tools for Windows PowerShell to create a new SSM document.

```powershell
$json = Get-Content C:\temp\JoinInstanceToDomain | Out-String
New-SSMDocument -Name JoinInstanceToDomain -Content $json -DocumentType Command
```

6. Run the following command in AWS Tools for Windows PowerShell to join the instance to the domain.

```powershell
Send-SSMCommand -InstanceId instance-id -DocumentName JoinInstanceToDomain
```

### Walkthrough: Manage parameters using hierarchies (AWS CLI)

This walkthrough shows how to work with parameters and parameter hierarchies by using the AWS CLI. For more information about parameter hierarchies, see Organizing parameters into hierarchies (p. 240).

#### To manage parameters using hierarchies

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create a parameter that uses the `allowedPattern` parameter and the `String` parameter type. The allowed pattern in this example means the value for the parameter must be between 1 and 4 digits long.

   **Linux**
   ```bash
   aws ssm put-parameter
   --name "/MyService/Test/MaxConnections" \
   --value 100 --allowed-pattern "\d{1,4}" \n   --type String
   ```

   **Windows**
   ```bash
   aws ssm put-parameter ^
   --name "/MyService/Test/MaxConnections" ^
   --value 100 --allowed-pattern "\d{1,4}" ^
   --type String
   ```
The command returns the version number of the parameter.

3. Run the following command to attempt to overwrite the parameter you just created with a new value.

Linux

```bash
aws ssm put-parameter
   --name "/MyService/Test/MaxConnections" \
   --value 10,000 \
   --type String \
   --overwrite
```

Windows

```bash
aws ssm put-parameter ^
   --name "/MyService/Test/MaxConnections" ^
   --value 10,000 ^
   --type String ^
   --overwrite
```

The system throws the following error because the new value does not meet the requirements of the allowed pattern you specified in the previous step.

```
An error occurred (ParameterPatternMismatchException) when calling the PutParameter operation: Parameter value, cannot be validated against allowedPattern: \d{1,4}
```

4. Run the following command to create a SecureString parameter that uses an AWS-managed customer master key (CMK). The allowed pattern in this example means the user can specify any character, and the value must be between 8 and 20 characters.

Linux

```bash
aws ssm put-parameter \
   --name "/MyService/Test/my-password" \
   --value "p#sW*rd33" \
   --allowed-pattern ".{8,20}" \
   --type SecureString
```

Windows

```bash
aws ssm put-parameter ^
   --name "/MyService/Test/my-password" ^
   --value "p#sW*rd33" ^
   --allowed-pattern ".{8,20}" ^
   --type SecureString
```

5. Run the following commands to create more parameters that use the hierarchy structure from the previous step.

Linux

```bash
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

Windows

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

6. Run the following command to get the value of two parameters.

Linux

aws ssm get-parameters \
   --names "/MyService/Test/user" "/MyService/Test/userType"

Windows

aws ssm get-parameters ^
   --names "/MyService/Test/user" "/MyService/Test/userType"

7. Run the following command to query for all parameters within a single level.

Linux

aws ssm get-parameters-by-path \
   --path "/MyService/Test"

Windows

aws ssm get-parameters-by-path ^
   --path "/MyService/Test"

8. Run the following command to delete two parameters
Linux

```bash
aws ssm delete-parameters \
   --names "/IADRegion/Dev/user" "/IADRegion/Dev/userType"
```

Windows

```bash
aws ssm delete-parameters ^
   --names "/IADRegion/Dev/user" "/IADRegion/Dev/userType"
```
AWS Systems Manager provides the following capabilities for taking action against or changing your AWS resources.

**Topics**
- AWS Systems Manager Automation (p. 278)
- AWS Systems Manager Change Calendar (p. 613)
- AWS Systems Manager Maintenance Windows (p. 620)

**AWS Systems Manager Automation**

Systems Manager Automation simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources. Automation enables you to do the following:

- Build Automation workflows to configure and manage instances and AWS resources.
- Create custom workflows or use pre-defined workflows maintained by AWS.
- Receive notifications about Automation tasks and workflows by using Amazon CloudWatch Events.
- Monitor Automation progress and execution details by using the Amazon EC2 or the AWS Systems Manager console.

**Primary Components**

AWS Systems Manager Automation uses the following components to run *automation workflows*.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation document (operational playbook)</td>
<td>A Systems Manager Automation document, or playbook, defines the Automation workflow (the actions that Systems Manager performs on your managed instances and AWS resources). Automation includes several pre-defined Automation documents that you can use to perform common tasks like restarting one or more EC2 instances or creating an Amazon Machine Image (AMI). You can create your own Automation documents as well. Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify. Steps run in sequential order. For more information, see <em>Working with Automation documents (Playbooks)</em> (p. 353).</td>
</tr>
</tbody>
</table>
Automation use cases

This section includes common uses cases for AWS Systems Manager Automation.

Perform common IT tasks

Automation can simplify common IT tasks such as changing the state of one or more instances (using an approval workflow) and managing instance states according to a schedule. Here are some examples:

- Use the AWS-StopEC2InstanceWithApproval document to request that one or more AWS Identity and Access Management (IAM) users approve the instance stop action. After the approval is received, Automation stops the instance.
- Use the AWS-StopEC2Instance document to automatically stop instances on a schedule by using Amazon CloudWatch Events or by using a maintenance window task. For example, you can configure an Automation workflow to stop instances every Friday evening, and then restart them every Monday morning.
- Use the AWS-UpdateCloudFormationStackWithApproval document to update resources that were deployed by using CloudFormation template. The update applies a new template. You can configure the Automation to request approval by one or more IAM users before the update begins.

For information about how to run an Automation workflow by using State Manager, see Running Automation workflows with triggers using State Manager (p. 319).

Safely perform disruptive tasks in bulk
Systems Manager includes features that help you target large groups of instances by using Amazon EC2 tags, and velocity controls that help you roll out changes according to the limits you define.

Use the AWS-RestartEC2InstanceWithApproval document to target an AWS resource group that includes multiple instances. You can configure the Automation workflow to use velocity controls. For example, you can specify the number of instances that should be restarted concurrently. You can also specify a maximum number of errors that are allowed before the Automation workflow is cancelled.

Simplify complex tasks

Automation offers one-click automations for simplifying complex tasks such as creating golden Amazon Machines Images (AMIs), and recovering unreachable EC2 instances. Here are some examples:

- Use the AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi documents to create golden AMIs from a source AMI. You can run custom scripts before and after updates are applied. You can also include or exclude specific packages from being installed. For examples of how to run these workflows, see Automation walkthroughs (p. 568).
- Use the AWSSupport-ExecuteEC2Rescue document to recover impaired instances. An instance can become unreachable for a variety of reasons, including network misconfigurations, RDP issues, or firewall settings. Troubleshooting and regaining access to the instance previously required dozens of manual steps before you could regain access. The AWSSupport-ExecuteEC2Rescue document lets you regain access by specifying an instance ID and clicking a button. For an example of how to run this workflow, see Walkthrough: Run the EC2Rescue tool on unreachable instances (p. 591).

Enhance operations security

Using delegated administration, you can restrict or elevate user permissions for various types of tasks.

Delegated administration enables you to provide permissions for certain tasks on certain resource without having to give a user direct permission to access the resources. This improves your overall security profile. For example, assume that User1 doesn't have permissions to restart EC2 instances, but you would like to authorize the user to do so. Instead of allowing User1 direct permissions, you can:

- Create an IAM role with the permissions required to successfully stop and start EC2 instances.
- Create an Automation document and embed the role in the document. (The easiest way to do this is to customize the AWS-RestartEC2Instance document and embed the role in the document instead of assigning an Automation service role [or assume role]).
- Modify IAM permissions for User1 and allow the user permission to run the document.

For an example of how to delegate access to an Automation workflow, see Running an Automation workflow by using delegated administration (p. 341).

Share best practices

Automation lets you share best practices with rest of your organization.

You can create best practices for resource management in Automation documents and easily share the documents across AWS Regions and groups. You can also constrain the allowed values for the parameters the document accepts.

Contents

- Getting started with Automation (p. 281)
- Working with Automation executions (p. 286)
- Working with Automation documents (Playbooks) (p. 353)
- Automation walkthroughs (p. 568)
- Troubleshooting Systems Manager Automation (p. 609)
Getting started with Automation

To set up Automation, you must verify user access to the Automation service and situationally configure roles so that the service can perform actions on your resources. To ensure proper access to Systems Manager Automation, review the following user and service role requirements.

Verifying user access for Automation workflows

Verify that you have permission to run Automation workflows. If your AWS Identity and Access Management (IAM) user account, group, or role is assigned administrator permissions, then you have access to Systems Manager Automation. If you don't have administrator permissions, then an administrator must give you permission by assigning the AmazonSSMFullAccess managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

**Important**
The IAM policy AmazonSSMFullAccess grants permissions to Systems Manager actions. However, some Automation documents, or playbooks, require permissions to other services, such as the document AWS-ReleaseElasticIP, which requires IAM permissions for ec2:ReleaseAddress. Therefore, you must review the actions taken in an Automation document to ensure your IAM user account, group, or role is assigned the necessary permissions to perform the actions included in the document.

Configuring a service role (assume role) access for Automation workflows

Automation workflows can be initiated under the context of a service role (or **assume role**). This allows the service to perform actions on your behalf. If you do not specify an assume role, Automation uses the context of the user who invoked the execution.

However, the following situations require that you specify a service role for Automation:

- When you want to restrict a user's privileges on a resource, but you want the user to run an Automation workflow that requires elevated privileges. In this scenario, you can create a service role with elevated privileges and allow the user to run the workflow.
- When you create a State Manager Association that runs an Automation workflow.
- When you have operations that you expect to run longer than 12 hours.
- When you are running an Automation document not owned by Amazon that uses the aws:executeScript action to call an AWS API operation or to act on an AWS resource. For information, see Permissions for running Automation executions (p. 358).

If you need to create a service role for Automation, you can use one of the following methods.

**Topics**

- Method 1: Use AWS CloudFormation to configure a service role for Automation (p. 281)
- Method 2: Use IAM to configure roles for Automation (p. 283)

**Method 1: Use AWS CloudFormation to configure a service role for Automation**

You can create a service role for Automation from an AWS CloudFormation template. After you create the service role, you can specify the service role in Automation workflows using the parameter AutomationAssumeRole. For information about how to run an Automation workflow using the Automation service role, see Running an Automation workflow by using an IAM service role (p. 337).
Create the service role using AWS CloudFormation

Use the following procedure to create the required IAM role for Systems Manager Automation by using AWS CloudFormation.

To create the required IAM role

1. Download the AWS-SystemsManager-AutomationServiceRole.zip folder. This folder includes the AWS-SystemsManager-AutomationServiceRole.yaml AWS CloudFormation template file.
3. Choose Create Stack.
4. In the Specify template section, choose Upload a template file.
5. Choose Browse, and then choose the AWS-SystemsManager-AutomationServiceRole.yaml AWS CloudFormation template file.
6. Choose Next.
7. On the Specify stack details page, in the Stack name field, enter a name.
8. On the Configure stack options page, you don't need to make any selections. Choose Next.
9. On the Review page, scroll down and choose the I acknowledge that AWS CloudFormation might create IAM resources option.
10. Choose Create.

AWS CloudFormation shows the CREATE_IN_PROGRESS status for approximately three minutes. The status changes to CREATE_COMPLETE after the stack is created and your roles are ready to use.

Important

If you run an automation workflow that invokes other services by using an AWS Identity and Access Management (IAM) service role, be aware that the service role must be configured with permission to invoke those services. This requirement applies to all AWS Automation documents (AWS-* documents) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RestartEC2Instance documents, to name a few. This requirement also applies to any custom Automation documents you create that invoke other AWS services by using actions that call other services. For example, if you use the aws:executeAwsApi, aws:createStack, or aws:copyImage actions, then you must configure the service role with permission to invoke those services. You can enable permissions to other AWS services by adding an IAM inline policy to the role. For more information, see (Optional) add an Automation inline policy to invoke other AWS services (p. 284).

Copy role information for Automation

Use the following procedure to copy information about the Automation service role from the AWS CloudFormation console. You must specify these roles when you run an Automation document.

Note

You do not need to copy role information using this procedure if you run the AWS-UpdateLinuxAmi or AWS-UpdateWindowsAmi documents. These documents already have the required roles specified as default values. The roles specified in these documents use IAM managed policies.

To copy the role names

2. Select the Automation **Stack name** you created in the previous procedure.
3. Choose the **Resources** tab.
4. Choose the **Physical ID** link for **AutomationServiceRole**. The IAM console opens to a summary of the Automation service role.
5. Copy the Amazon Resource Name (ARN) next to **Role ARN**. The ARN is similar to the following:
   `arn:aws:iam::12345678:role/AutomationServiceRole`
6. Paste the ARN into a text file to use later.

You have finished configuring the service role for Automation. You can now use the Automation service role ARN in your Automation documents.

**Method 2: Use IAM to configure roles for Automation**

If you need to create a service role for Systems Manager Automation, complete the following tasks. For more information on when a service role is required for Automation, see **Getting started with Automation (p. 281)**.

**Tasks**
- Task 1: Create a service role for Automation (p. 283)
- Task 2: Attach the iam:PassRole policy to your Automation role (p. 285)
- Task 3: Configure user access to Automation (p. 285)

**Task 1: Create a service role for Automation**

Use the following procedure to create a service role (or **assume role**) for Systems Manager Automation.

**Note**
You can also use this role in Automation documents, such as the **AWS-CreateManagedLinuxInstance** document. Using this role or the ARNs in Automation documents enables Automation to perform actions in your environment, such as launch new instances and perform actions on your behalf.

**To create an IAM role and allow Automation to assume it**

2. In the navigation pane, choose **Roles**, and then choose **Create role**.
3. Under **Select type of trusted entity**, choose **AWS service**.
4. In the **Choose a use case** section, choose **Systems Manager**, and then choose **Next: Permissions**.
5. On the **Attached permissions policy** page, search for the **AmazonSSMAutomationRole** policy, choose it, and then choose **Next: Review**.
6. On the **Review** page, type a name in the **Role name** box, and then type a description.
7. Choose **Create role**. The system returns you to the **Roles** page.
8. On the **Roles** page, choose the role you just created to open the **Summary** page. Note the **Role Name** and **Role ARN**. You will specify the role ARN when you attach the **iam:PassRole** policy to your IAM account in the next procedure. You can also specify the role name and the ARN in Automation documents.

**Note**
The AmazonSSMAutomationRole policy assigns the Automation role permission to a subset of AWS Lambda functions within your account. These functions begin with "Automation". If you plan to use Automation with Lambda functions, the Lambda ARN must use the following format:
```
"arn:aws:lambda:*:*:function:Automation"
```
If you have existing Lambda functions whose ARNs do not use this format, then you must also attach an additional Lambda policy to your automation role, such as the **AWSLambdaRole** policy. The additional policy or role must provide broader access to Lambda functions within the AWS account.

**Optional** add an Automation inline policy to invoke other AWS services

If you run an automation that invokes other AWS services by using an IAM service role, the service role must be configured with permission to invoke those services. This requirement applies to all AWS Automation documents (AWS-* documents) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RestartEC2Instance documents, to name a few. This requirement also applies to any custom Automation documents you create that invoke other AWS services by using actions that call other services. For example, if you use the aws:executeAwsApi, aws:CreateStack, or aws:copyImage actions, to name a few, then you must configure the service role with permission to invoke those services. You can enable permissions to other AWS services by adding an IAM inline policy to the role.

**To embed an inline policy for a service role (IAM console)**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles**.
3. In the list, choose the name of the role that you want to edit.
4. Choose the **Permissions** tab.
5. Choose **Add inline policy**.
6. Choose the **JSON** tab.
7. Enter a JSON policy document for the AWS services you want to invoke. Here are two example JSON policy documents.

**Amazon S3 PutObject and GetObject Example**

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

**Amazon EC2 CreateSnapshot and DescribeSnapShots Example**

```json
{
   "Version":"2012-10-17",
   "Statement":[
   {
      "Effect":"Allow",
      "Action":"ec2:CreateSnapshot",
      "Resource":"*"
   },
   {
      "Effect":"Allow",
      "Action":"ec2:DescribeSnapshots",
```
Getting started with Automation

For details about the IAM policy language, see IAM JSON Policy Reference in the IAM User Guide.

8. When you are finished, choose Review policy. The Policy Validator reports any syntax errors.
9. On the Review policy page, enter a Name for the policy that you are creating. Review the policy Summary to see the permissions that are granted by your policy. Then choose Create policy to save your work.
10. After you create an inline policy, it is automatically embedded in your role.

**Task 2: Attach the iam:PassRole policy to your Automation role**

Use the following procedure to attach the iam:PassRole policy to your Automation service role. This enables the Automation service to pass the role to other services or Systems Manager capabilities when running Automation workflows.

**To attach the iam:PassRole policy to your Automation role**

1. In the Summary page for the role you just created, choose the Permissions tab.
2. Choose Add inline policy.
3. On the Create policy page, choose the Visual editor tab.
4. Choose Service, and then choose IAM.
5. Choose Select actions.
6. In the Filter actions text box, type PassRole, and then choose the PassRole option.
7. Choose Resources. Verify that Specific is selected, and then choose Add ARN.
8. In the Specify ARN for role field, paste the Automation role ARN that you copied at the end of Task 1. The system populates the Account and Role name with path fields.

**Note**
If you want the Automation service role to attach an IAM instance profile role to an EC2 instance, then you must add the ARN of the IAM instance profile role. This allows the Automation service role to pass the IAM instance profile role to the target EC2 instance.

11. On the Review Policy page, type a name and then choose Create Policy.

**Task 3: Configure user access to Automation**

If your AWS Identity and Access Management (IAM) user account, group, or role is assigned administrator permissions, then you have access to Systems Manager Automation. If you don’t have administrator permissions, then an administrator must give you permission by assigning the AmazonSSMFullAccess managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

Use the following procedure to configure a user account to use Automation. The user account you choose will have permission to configure and run Automation. If you need to create a new user account, see Creating an IAM User in Your AWS Account in the IAM User Guide.

**To configure user access and attach the iam:PassRole policy to a user account**

1. In the IAM navigation pane, choose Users, and then choose the user account you want to configure.
2. On the Permissions tab, in the policies list, verify that either the AmazonSSMFullAccess policy is listed or there is a comparable policy that gives the account permissions to access Systems Manager.
3. Choose **Add inline policy**.
4. On the **Create policy** page, choose **Visual Editor**, and then choose **Choose a service**.
5. From **AWS Services**, choose **AWS Identity and Access Management**.
6. For **Actions**, enter **PassRole** in the **Filter actions** prompt, and choose **PassRole**.
7. In the **Resources** section, choose **Add ARN**, paste the ARN for the Automation service role you copied at the end of Task 1, and then choose **Add**.
8. Choose **Review policy**.
9. On the **Review Policy** page, provide a **Name** for the policy and then choose **Create policy**.

You have finished configuring the required roles for Automation. You can now use the Automation service role ARN in your Automation documents.

**Working with Automation executions**

This section includes information about how to run Systems Manager Automation workflows. For more examples of how to run Automation workflows, see Automation walkthroughs (p. 568).

**Contents**
- Running a simple Automation workflow (p. 286)
- Running an Automation workflow manually (p. 290)
- Running an Automation workflow with approvers (p. 296)
- Running Automation workflows that use targets and rate controls (p. 299)
- Running Automation workflows based on triggers (p. 314)
- Running Automation workflows by using different security models (p. 333)
- Running Automation workflows in multiple AWS Regions and accounts (p. 344)

**Running a simple Automation workflow**

The following procedures describe how to run a simple Systems Manager Automation workflow using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell. The workflow runs in the context of the current AWS Identity and Access Management (IAM) user. This means that you don't need to configure additional IAM permissions as long as you have permission to run the Automation document, or playbook, and any actions called by the document. If you have administrator permissions in IAM, then you already have permission to run this Automation workflow.

**Note**
For information about how to run an Automation workflow that uses an IAM service role or more advanced forms of delegated administration, see Running Automation workflows by using different security models (p. 333).

**Running a simple Automation workflow (console)**

The following procedure describes how to use the Systems Manager console to run a simple Automation workflow.

**To run a simple Automation workflow**
2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a document. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a document
that you own, choose the **Owned by me** tab. To view a document that is shared with your account, choose the **Shared with me** tab. To view all documents, choose the **All documents** tab.

**Note**
You can view information about a document by choosing the document name.

4. In the **Document details** section, verify that **Document version** is set to the version that you want to run. The system includes the following version options:

- **Default version at runtime**: Choose this option if the Automation document is updated periodically and a new default version is assigned.
- **Latest version at runtime**: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.
- **1 (Default)**: Choose this option to run the first version of the document, which is the default.

5. Choose **Next**.

6. In the **Execution Mode** section, choose **Simple execution**.

7. In the **Input parameters** section, specify the required inputs. Optionally, you can choose an IAM service role from the **AutomationAssumeRole** list.

8. Choose **Execute**.

The console displays the status of the Automation execution. If the Automation fails to run, see *Troubleshooting Systems Manager Automation (p. 609).*

**Running a simple Automation workflow (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to run a simple Automation workflow.

**To run a simple Automation workflow**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see *Install or upgrade AWS command line tools (p. 57).*

2. Run the following command to start a simple Automation workflow.

   **Linux**
   ```
   aws ssm start-automation-execution \
   --document-name DocumentName \
   --parameters ParametersRequiredByDocument
   ```

   **Windows**
   ```
   aws ssm start-automation-execution ^
   --document-name DocumentName ^
   --parameters ParametersRequiredByDocument
   ```

   **PowerShell**
   ```
   Start-SSMAutomationExecution \
   -DocumentName DocumentName \
   -Parameter ParametersRequiredByDocument
   ```

   Here is an example using the document **AWS-RestartEC2Instance** to restart the specified EC2 instance.
Working with Automation executions

**Linux**

```bash
aws ssm start-automation-execution \
  --document-name "AWS-RestartEC2Instance" \
  --parameters "InstanceId=i-1234567890abcdef0"
```

**Windows**

```bash
aws ssm start-automation-execution ^
  --document-name "AWS-RestartEC2Instance" ^
  --parameters "InstanceId=i-1234567890abcdef0"
```

**PowerShell**

```powershell
Start-SSMAutomationExecution `\
  -DocumentName AWS-RestartEC2Instance `\
  -Parameter @{"InstanceId"="i-1234567890abcdef0"}
```

The system returns information like the following.

**Linux**

```json
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

**Windows**

```json
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

**PowerShell**

```
4105a4fc-f944-11e6-9d32-0123456789ab
```

3. Run the following command to retrieve the status of the Automation workflow.

**Linux**

```bash
aws ssm describe-automation-executions \
  --filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

**Windows**

```bash
aws ssm describe-automation-executions ^
  --filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

**PowerShell**

```powershell
Get-SSMAutomationExecutionList | `\
  Where {$_ AutomationExecutionId -eq "4105a4fc-f944-11e6-9d32-0123456789ab"}
```
The system returns information like the following.

**Linux**

```json
{
    "AutomationExecutionMetadataList": [
        {
            "AutomationExecutionStatus": "InProgress",
            "CurrentStepName": "stopInstances",
            "Outputs": {},
            "DocumentName": "AWS-RestartEC2Instance",
            "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab",
            "DocumentVersion": "1",
            "ResolvedTargets": {
                "ParameterValues": [],
                "Truncated": false
            },
            "AutomationType": "Local",
            "Mode": "Auto",
            "ExecutionStartTime": 1564600648.159,
            "CurrentAction": "aws:changeInstanceState",
            "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
            "LogFile": "",
            "Targets": []
        }
    ]
}
```

**Windows**

```json
{
    "AutomationExecutionMetadataList": [
        {
            "AutomationExecutionStatus": "InProgress",
            "CurrentStepName": "stopInstances",
            "Outputs": {},
            "DocumentName": "AWS-RestartEC2Instance",
            "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab",
            "DocumentVersion": "1",
            "ResolvedTargets": {
                "ParameterValues": [],
                "Truncated": false
            },
            "AutomationType": "Local",
            "Mode": "Auto",
            "ExecutionStartTime": 1564600648.159,
            "CurrentAction": "aws:changeInstanceState",
            "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
            "LogFile": "",
            "Targets": []
        }
    ]
}
```

**PowerShell**

```
<table>
<thead>
<tr>
<th>AutomationExecutionId</th>
<th>4105a4fc-f944-11e6-9d32-0123456789ab</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutomationExecutionStatus</td>
<td>InProgress</td>
</tr>
</tbody>
</table>
```
### Running an Automation workflow manually

The following procedures describe how to use the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell to run a Systems Manager Automation workflow using the manual execution mode. By using the manual execution mode, the Automation workflow starts in a *Waiting* status and pauses in the *Waiting* status between each step. This allows you to control when the workflow proceeds, which is useful if you need to review the result of a step before continuing.

The workflow runs in the context of the current AWS Identity and Access Management (IAM) user. This means that you don't need to configure additional IAM permissions as long as you have permission to run the Automation document, or playbook, and any actions called by the document. If you have administrator permissions in IAM, then you already have permission to run this Automation workflow.

**Note**
For information about how to run an Automation workflow that uses an IAM service role or more advanced forms of delegated administration, see Running Automation workflows by using different security models (p. 333).

### Running an Automation workflow step by step (console)

The following procedure shows how to use the Systems Manager console to manually run an Automation workflow step by step.

**To run an Automation workflow step by step**

2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a document. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a document that you own, choose the **Owned by me** tab. To view a document that is shared with your account, choose the **Shared with me** tab. To view all documents, choose the **All documents** tab.

   **Note**
   You can view information about a document by choosing the document name.

4. In the **Document details** section, verify that **Document version** is set to the version that you want to run. The system includes the following version options:
• **Default version at runtime**: Choose this option if the Automation document is updated periodically and a new default version is assigned.

• **Latest version at runtime**: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.

• **1 (Default)**: Choose this option to run the first version of the document, which is the default.

5. Choose **Next**.

6. In the **Execution Mode** section, choose **Manual execution**.

7. In the **Input parameters** section, specify the required inputs. Optionally, you can choose an IAM service role from the **AutomationAssumeRole** list.

8. Choose **Execute**.

9. Choose **Execute this step** when you are ready to start the first step of the Automation workflow. The Automation workflow proceeds with step one and pauses before running any subsequent steps specified in the Automation document you chose in step 3 of this procedure. If the document has multiple steps, you must select **Execute this step** for each step for the workflow to proceed.

   **Note**
   The console displays the status of the Automation execution. If the Automation fails to run a step, see Troubleshooting Systems Manager Automation (p. 609).

10. After you complete all steps specified in the Automation document, choose **Complete and view results** to finish the Automation workflow and view the results.

**Running an Automation workflow step by step (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to manually run an Automation workflow step by step.

**To run an Automation workflow step by step**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to start a manual Automation workflow:

   **Linux**
   ```bash
   aws ssm start-automation-execution \
   --document-name DocumentName \
   --mode Interactive \
   --parameters ParametersRequiredByDocument
   ```

   **Windows**
   ```bash
   aws ssm start-automation-execution ^
   --document-name DocumentName ^
   --mode Interactive ^
   --parameters ParametersRequiredByDocument
   ```

   **PowerShell**
   ```bash
   Start-SSMAutomationExecution `^`
   -DocumentName DocumentName `^`
   -Mode Interactive `^`
   -Parameter ParametersRequiredByDocument
   ```
Here is an example using the document AWS-RestartEC2Instance to restart the specified EC2 instance.

Linux

```bash
aws ssm start-automation-execution \
--document-name "AWS-RestartEC2Instance" \
--mode Interactive \
--parameters "InstanceId=i-1234567890abcdef0"
```

Windows

```bash
aws ssm start-automation-execution ^
--document-name "AWS-RestartEC2Instance" ^
--mode Interactive ^
--parameters "InstanceId=i-1234567890abcdef0"
```

PowerShell

```powershell
Start-SSMAutomationExecution `^
-DocumentName AWS-RestartEC2Instance `^
-Mode Interactive `^
-Parameter @{"InstanceId"="i-1234567890abcdef0"}
```

The system returns information like the following.

Linux

```json
{
  "AutomationExecutionId": "ba9cd881-1b36-4d31-a698-0123456789ab"
}
```

Windows

```json
{
  "AutomationExecutionId": "ba9cd881-1b36-4d31-a698-0123456789ab"
}
```

PowerShell

```powershell
27ba8174-59ae-4e13-8626-0123456789ab
```

3. Run the following command when you are ready to start the first step of the Automation workflow. The Automation workflow proceeds with step one and pauses before running any subsequent steps specified in the Automation document you chose in step 1 of this procedure. If the document has multiple steps, you must run the following command for each step for the workflow to proceed.

Linux

```bash
aws ssm send-automation-signal \
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab \
--signal-type StartStep \
--payload StepName="stopInstances"
```
Windows

```bash
aws ssm send-automation-signal ^
   --automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab ^
   --signal-type StartStep ^
   --payload StepName="stopInstances"
```

PowerShell

```powershell
Send-SSMAutomationSignal `^`  
   -AutomationExecutionId 27ba8174-59ae-4e13-8626-0123456789ab `^`  
   -SignalType StartStep `^`  
   -Payload @{"StepName=","stopInstances"} `^`
```

There is no output if the command succeeds.

4. Run the following command to retrieve the status of each step execution in the Automation workflow.

Linux

```bash
aws ssm describe-automation-step-executions \  
   --automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab
```

Windows

```bash
aws ssm describe-automation-step-executions ^  
   --automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab
```

PowerShell

```powershell
Get-SSMAutomationStepExecution `^`  
   -AutomationExecutionId 27ba8174-59ae-4e13-8626-e177cdcc1686 `^`
```

The system returns information like the following.

Linux

```json
{
   "StepExecutions": [
      {
         "StepName": "stopInstances",
         "Action": "aws:changeInstanceState",
         "ExecutionStartTime": 1557167178.42,
         "ExecutionEndTime": 1557167220.617,
         "StepStatus": "Success",
         "Inputs": {
            "DesiredState": "stopped",
            "InstanceIds": "["i-1234567890abcdef0"]"
         },
         "Outputs": {
            "InstanceStates": ["stopped"
         ],
         "StepExecutionId": "654243ba-71e3-4771-b04f-0123456789ab",
```
"OverriddenParameters": {},
"ValidNextSteps": [
  "startInstances"
],
{
  "StepName": "startInstances",
  "Action": "aws:changeInstanceState",
  "ExecutionStartTime": 1557167273.754,
  "ExecutionEndTime": 1557167480.73,
  "StepStatus": "Success",
  "Inputs": {
    "DesiredState": "\"running\"",
    "InstanceIds": "[\"i-1234567890abcdef0\"]"
  },
  "Outputs": {
    "InstanceStates": [
      "running"
    ]
  },
  "StepExecutionId": "8a4a1e0d-dc3e-4039-a599-0123456789ab",
  "OverriddenParameters": {}
}
}

Windows

{
  "StepExecutions": [
    {
      "StepName": "stopInstances",
      "Action": "aws:changeInstanceState",
      "ExecutionStartTime": 1557167178.42,
      "ExecutionEndTime": 1557167220.617,
      "StepStatus": "Success",
      "Inputs": {
        "DesiredState": "\"stopped\"",
        "InstanceIds": "[\"i-1234567890abcdef0\"]"
      },
      "Outputs": {
        "InstanceStates": [
          "stopped"
        ]
      },
      "StepExecutionId": "654243ba-71e3-4771-b04f-0123456789ab",
      "OverriddenParameters": {},
      "ValidNextSteps": [
        "startInstances"
      ]
    },
    {
      "StepName": "startInstances",
      "Action": "aws:changeInstanceState",
      "ExecutionStartTime": 1557167273.754,
      "ExecutionEndTime": 1557167480.73,
      "StepStatus": "Success",
      "Inputs": {
        "DesiredState": "\"running\"",
        "InstanceIds": "[\"i-1234567890abcdef0\"]"
      },
      "Outputs": {
        "InstanceStates": [
          "running"
        ]
      }
    }
  ]
}
PowerShell

Action : aws:changeInstanceState
ExecutionEndTime : 5/6/2019 19:45:46
ExecutionStartTime : 5/6/2019 19:45:03
FailureDetails :
FailureMessage :
Inputs : {
[DesiredState, "stopped"],
[InstanceIds,
"i-1234567890abcdef0"]
}
IsCritical : False
IsEnd : False
MaxAttempts : 0
NextStep :
OnFailure :
Outputs : {
InstanceStates,
}
OverriddenParameters : {}
Response :
ResponseCode :
StepExecutionId : 8fcc9641-24b7-40b3-a9be-0123456789ab
StepName : stopInstances
StepStatus : Success
TimeoutSeconds : 0
ValidNextSteps : {startInstances}

5. Run the following command to complete the Automation workflow after all steps specified within the chosen Automation document have finished.

Linux

aws ssm stop-automation-execution \
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab \
--type Complete

Windows

aws ssm stop-automation-execution ^
--automation-execution-id ba9cd881-1b36-4d31-a698-0123456789ab ^
--type Complete

PowerShell

Stop-SSMAutomationExecution `
-AutomationExecutionId 27ba8174-59ae-4e13-8626-0123456789ab ` 
-Type Complete

There is no output if the command succeeds.
Running an Automation workflow with approvers

The following procedures describe how to use the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell to run an AWS Systems Manager Automation workflow with approvals using simple execution. The workflow uses the Automation action `aws:approve`, which temporarily pauses the Automation workflow until the designated principals either approve or deny the action. The Automation workflow runs in the context of the current AWS Identity and Access Management (IAM) user. This means that you don't need to configure additional IAM permissions as long as you have permission to run the Automation document, or playbook, and any actions called by the document. If you have administrator permissions in IAM, then you already have permission to run this Automation workflow.

**Note**
For information about how to run an Automation workflow that uses an IAM service role or more advanced forms of delegated administration, see Running Automation workflows by using different security models (p. 333).

**Before You Begin**

In addition to the standard inputs required by the Automation document, the `aws:approve` action requires the following two parameters:

- A list of approvers. The list of approvers must contain at least one approver in the form of an IAM user or a user ARN. If multiple approvers are provided, a corresponding minimum approval count must also be specified within the automation document.
- An Amazon Simple Notification Service (Amazon SNS) topic ARN. The Amazon SNS topic name must start with `Automation`.

This procedure assumes that you have already created an Amazon SNS topic, which is required to deliver the approval request. For information, see Create a Topic in the Amazon Simple Notification Service Developer Guide.

**Running an Automation workflow with approvers (console)**

**To run an Automation workflow with approvers**

The following procedure describes how to use the Systems Manager console to run an Automation workflow with approvers.

2. In the navigation pane, choose Automation, and then choose Execute automation.
3. In the Automation document list, choose a document. Choose one or more options in the Document categories pane to filter SSM documents according to their purpose. To view a document that you own, choose the Owned by me tab. To view a document that is shared with your account, choose the Shared with me tab. To view all documents, choose the All documents tab.

**Note**
You can view information about a document by choosing the document name.

4. In the Document details section, verify that Document version is set to the version that you want to run. The system includes the following version options:

   - **Default version at runtime**: Choose this option if the Automation document is updated periodically and a new default version is assigned.
   - **Latest version at runtime**: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.
   - **1 (Default)**: Choose this option to run the first version of the document, which is the default.
5. Choose Next.
7. In the Input parameters section, specify the required input parameters.
   
   For example, if you chose the AWS-StartEC2InstanceWithApproval document, then you must specify or choose instance IDs for the InstanceId parameter.
8. In the Approvers section, specify the IAM users or user ARNs of approvers for the automation action.
9. In the SNSTopicARN section, specify the SNS topic ARN to use for sending approval notification. The SNS topic name must start with Automation.
10. Optionally, you can choose an IAM service role from the AutomationAssumeRole list.
11. Choose Execute automation.

The specified approver receives an Amazon SNS notification with details to approve or reject the Automation workflow. This approval action is valid for 7 days from the date of issue and can be issued using the Systems Manager console or the AWS Command Line Interface (AWS CLI).

If you chose to approve the Automation workflow, the workflow continues to run the steps included in the specified Automation document. The console displays the status of the Automation execution. If the Automation fails to run, see Troubleshooting Systems Manager Automation (p. 609).

To approve or deny an Automation workflow

2. In the navigation pane, choose Automation, and then select the Automation workflow that was run in the previous procedure.
3. Choose Actions and then choose Approve/Deny.
4. Choose to Approve or Deny and optionally provide a comment.
5. Choose Submit.

Running an Automation workflow with approvers (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to run an Automation workflow with approvers.

To run an Automation workflow with approvers

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).
2. Use the following command to run an Automation workflow with approvers. In the document name section, specify an Automation document that includes the Automation action, aws:approve.

   For Approvers, specify the IAM users or user ARNs of approvers for the action. For SNSTopic, specify the SNS topic ARN to use to send approval notification. The SNS topic name must start with Automation.

   Note
   
   The specific names of the parameter values for approvers and the SNS topic depend on the values specified within the document you choose.

   Linux

   ```bash
   aws ssm start-automation-execution \ 
   --document-name "AWS-StartEC2InstanceWithApproval" \ 
   --input-parameters "InstanceId=instance1,instance2" \ 
   --approver-arns "arn:aws:iam::123456789012:user/Approver1,arn:aws:iam::123456789012:user/Approver2" \ 
   --snstopic-arn "arn:aws:sns:us-east-1:123456789012:ApprovalTopic" \ 
   --automation-assume-role-arn "arn:aws:iam::123456789012:role/AutomationRole" \ 
   --region us-east-1
   ```
Windows

aws ssm start-automation-execution
   --document-name "AWS-StartEC2InstanceWithApproval" 
   --parameters
   "InstanceId=i-1234567890abcdef0,Approvers=arn:aws:iam::123456789012:role/Administrator,SNSTopicArn=arn:aws:sns:us-east-1:123456789012:AutomationApproval"

PowerShell

Start-SSMAutomationExecution
   -DocumentName AWS-StartEC2InstanceWithApproval
   -Parameters @{
   "InstanceId"="i-1234567890abcdef0"
   "Approvers"="arn:aws:iam::123456789012:role/Administrator"
   "SNSTopicArn"="arn:aws:sns:us-east-1:123456789012:AutomationApproval"
}

The system returns information like the following.

Linux

{
   "AutomationExecutionId": "df325c6d-b1b1-4aa0-8003-6cb7338213c6"
}

Windows

{
   "AutomationExecutionId": "df325c6d-b1b1-4aa0-8003-6cb7338213c6"
}

PowerShell

462fa82a-7fff-430a-8490-0123456789ab

To approve an Automation workflow

- Run the following command to approve an Automation workflow.

Linux

aws ssm send-automation-signal
   --automation-execution-id "4105a4fc-f944-11e6-9d32-0123456789ab"
   --signal-type "Approve"
   --payload "Comment=Replace_This_With_Approve_Comment"

Windows

aws ssm send-automation-signal
   --automation-execution-id "4105a4fc-f944-11e6-9d32-0123456789ab"
--signal-type "Approve" ^
--payload "Comment=Replace_This_With_Approve_Comment"

PowerShell

Send-SSMAutomationSignal `
-AutomationExecutionId 462fa82a-7fff-430a-8490-0123456789ab ` 
-SignalType Approve ` 
-Payload @{"Comment"="Replace_This_With_Approval_Comment"}

There is no output if the command succeeds.

**To deny an Automation workflow**

- Run the following command to deny an Automation workflow.

  **Linux**

  ```bash
  aws ssm send-automation-signal \
  --automatic-execution-id "4105a4fc-f944-11e6-9d32-0123456789ab" \ 
  --signal-type "Deny" \ 
  --payload "Comment=Replace_This_With_Deny_Comment"
  ```

  **Windows**

  ```bash
  aws ssm send-automation-signal ^
  --automatic-execution-id "4105a4fc-f944-11e6-9d32-0123456789ab" ^
  --signal-type "Deny" ^
  --payload "Comment=Replace_This_With_Deny_Comment"
  ```

  **PowerShell**

  ```powershell
  Send-SSMAutomationSignal ` 
  -AutomationExecutionId 462fa82a-7fff-430a-8490-0123456789ab ` 
  -SignalType Deny ` 
  -Payload @{"Comment"="Replace_This_With_Deny_Comment"}
  ```

  There is no output if the command succeeds.

**Running Automation workflows that use targets and rate controls**

AWS Systems Manager enables you to run Automation workflows on a fleet of AWS resources by using targets. Additionally, you can control the execution of the Automation across your fleet by specifying a concurrency value and an error threshold. The concurrency value determines how many resources are allowed to run the Automation simultaneously. An error threshold determines how many Automation executions are allowed to fail before Systems Manager stops sending the workflow to other resources. The concurrency and error threshold features are collectively called *rate controls*.

For more information about concurrency and error thresholds, see About concurrency and error thresholds (p. 313). For more information about targets, see About targets (p. 308).
The following procedures describe how to run an Automation workflow with targets and rate controls by using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell.

Running an Automation workflow with targets and rate controls (console)

The following procedure describes how to use the Systems Manager console to run an Automation workflow with targets and rate controls.

To run an Automation workflow with targets and rate controls

2. In the navigation pane, choose Automation, and then choose Execute automation.
3. In the Automation document list, choose a document. Choose one or more options in the Document categories pane to filter SSM documents according to their purpose. To view a document that you own, choose the Owned by me tab. To view a document that is shared with your account, choose the Shared with me tab. To view all documents, choose the All documents tab.
   Note
   You can view information about a document by choosing the document name.
4. In the Document details section, verify that Document version is set to the version that you want to run. The system includes the following version options:
   • Default version at runtime: Choose this option if the Automation document is updated periodically and a new default version is assigned.
   • Latest version at runtime: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.
   • 1 (Default): Choose this option to run the first version of the document, which is the default.
5. Choose Next.
6. In the Execution Mode section, choose Rate Control. You must use this mode or Multi-account and Region if you want to use targets and rate controls.
7. In the Targets section, choose how you want to target the AWS Resources where you want to run the Automation. These options are required.
   a. Use the Parameter list to choose a parameter. The items in the Parameter list are determined by the parameters in the Automation document that you selected at the start of this procedure. By choosing a parameter you define the type of resource on which the Automation workflow runs.
   b. Use the Targets list to choose how you want to target resources.
      i. If you chose to target resources by using parameter values, then enter the parameter value for the parameter you chose in the Input parameters section.
      ii. If you chose to target resources by using AWS Resource Groups, then choose the name of the group from the Resource Group list.
      iii. If you chose to target resources by using tags, then enter the tag key and (optionally) the tag value in the fields provided. Choose Add.
      iv. If you want to run an Automation playbook on all instances in the current AWS account and Region, then choose All instances.
8. In the Input parameters section, specify the required inputs. Optionally, you can choose an IAM service role from the AutomationAssumeRole list.
   Note
   You may not need to choose some of the options in the Input parameters section. This is because you targeted resources by using tags or a resource group. For example, if you chose the AWS-RestartEC2Instance document, then you don’t need to specify or choose instance
IDs in the *Input parameters* section. The Automation execution locates the instances to restart by using the tags or Resource Group you specified.

9. Use the options in the *Rate control* section to restrict the number of AWS resources that can run the Automation within each account-Region pair.

   In the *Concurrency* section, choose an option:
   - Choose *targets* to enter an absolute number of targets that can run the Automation workflow simultaneously.
   - Choose *percentage* to enter a percentage of the target set that can run the Automation workflow simultaneously.

10. In the *Error threshold* section, choose an option:
   - Choose *errors* to enter an absolute number of errors allowed before Automation stops sending the workflow to other resources.
   - Choose *percentage* to enter a percentage of errors allowed before Automation stops sending the workflow to other resources.

11. Choose *Execute*.

**Running an Automation workflow with targets and rate controls (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to run an Automation workflow with targets and rate controls.

**To run an Automation workflow with targets and rate controls**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.
   
   For information, see [Install or upgrade AWS command line tools](p. 57).

2. Run the following command to view a list of documents.

   Linux
   ```bash
   aws ssm list-documents
   ```

   Windows
   ```cmd
   aws ssm list-documents
   ```

   PowerShell
   ```powershell
   Get-SSMDocumentList
   ```

   Note the name of the Automation document that you want to run.

3. Run the following command to view details about the Automation document. Note a parameter name (for example, `InstanceId`) that you want to use for the `--target-parameter-name` option. This parameter determines the type of resource on which the Automation runs.

   Linux
   ```bash
   aws ssm describe-document \
   --name document_name
   ```
4. Create a command that uses the targets and rate control options you want to run. Here are some template commands to help.

**Targeting using tags**

**Linux**

```
aws ssm start-automation-execution \
  --document-name document_name \
  --targets Key=tag:key_name,Values=value \
  --target-parameter-name parameter_name \
  --parameters "input_parameter_name1=input_parameter_value1,input_parameter_name2=input_parameter_value2" \
  --max-concurrency 10 \
  --max-errors 25%
```

**Windows**

```
aws ssm start-automation-execution ^ \
  --document-name document_name ^ \
  --targets Key=tag:key_name,Values=value ^ \
  --target-parameter-name parameter_name ^ \
  --parameters "input_parameter_name1=input_parameter_value1,input_parameter_name2=input_parameter_value2" ^ \
  --max-concurrency 10 ^ \
  --max-errors 25%
```

**PowerShell**

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag:key_name"
$Targets.Values = "value"

Start-SSMAutomationExecution ` 
-DocumentName "DocumentName" ` 
-Targets $Targets ` 
-TargetParameterName "Target_Parameter" ` 
-Parameter @{"input_parameter_name1"="input_parameter_value1";"input_parameter_name2"="input_parameter_value2"} ` 
-MaxConcurrency "a_number_of_instances_or_a_percentage_of_target_set" ` 
-MaxError "a_number_of_errors_or_a_percentage_of_target_set"
```

**Targeting using parameter values**

...
Working with Automation executions

Linux

```bash
aws ssm start-automation-execution \
  --document-name document_name \
  --targets Key=ParameterValues,Values=value_1,value_2,value_3 \
  --target-parameter-name parameter_name \
  --parameters "input_parameter_name1=input_parameter_value1" \
  --max-concurrency 10 \
  --max-errors 25%
```

Windows

```bash
aws ssm start-automation-execution ^
  --document-name document_name ^
  --targets Key=ParameterValues,Values=value_1,value_2,value_3 ^
  --target-parameter-name parameter_name ^
  --parameters "input_parameter_name1=input_parameter_value1" ^
  --max-concurrency 10 ^
  --max-errors 25%
```

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ParameterValues"
$Targets.Values = "value_1","value_2","value_3"

Start-SSMAutomationExecution `\n  -DocumentName "DocumentName" `\n  -Targets $Targets `\n  -TargetParameterName "Parameter_Name" `\n  -Parameter @{"input_parameter_name1"="input_parameter_value1"} `\n  -MaxConcurrency "a_number_of_instances_or_a_percentage_of_target_set" `\n  -MaxError "a_number_of_errors_or_a_percentage_of_target_set"
```

Targeting using AWS Resource Groups

Linux

```bash
aws ssm start-automation-execution \
  --document-name document_name \
  --targets Key=ResourceGroup,Values=Resource_Group_name \
  --target-parameter-name parameter_name \
  --parameters "input_parameter_name1=input_parameter_value1,input_parameter_name2=input_parameter_value2" \
  --max-concurrency 10 \
  --max-errors 25%
```

Windows

```bash
aws ssm start-automation-execution ^
  --document-name document_name ^
  --targets Key=ResourceGroup,Values=Resource_Group_name ^
  --target-parameter-name parameter_name ^
  --parameters "input_parameter_name1=input_parameter_value1,input_parameter_name2=input_parameter_value2" ^
  --max-concurrency 10 ^
```
---max-errors 25%

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "Resource_Group_Name"

Start-SSMAutomationExecution
 -DocumentName "DocumentName"
 -Targets $Targets
 -TargetParameterName "Parameter_Name"
 @{"input_parameter_name1"="input_parameter_value1";"input_parameter_name2"="input_parameter_value2";
 -MaxConcurrency "a_number_of_instances_or_a_percentage_of_target_set"
 -MaxError "a_number_of_errors_or_a_percentage_of_target_set"
```

Targeting all instances in the current AWS account and Region

Linux

```
aws ssm start-automation-execution \
 --document-name document_name \
 --targets "Key=AWS::EC2::Instance,Values="*" \
 --target-parameter-name instanceId \
 --parameters "input_parameter_name1"="input_parameter_value1" \
 --max-concurrency 10 \
 --max-errors 25%
```

Windows

```
aws ssm start-automation-execution ^
 --document-name document_name ^
 --targets Key=AWS::EC2::Instance,Values=* ^
 --target-parameter-name instanceId ^
 --parameters "input_parameter_name1"="input_parameter_value1" ^
 --max-concurrency 10 ^
 --max-errors 25%
```

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "AWS::EC2::Instance"
$Targets.Values = "*"

Start-SSMAutomationExecution \
 -DocumentName "DocumentName"
 -Targets $Targets
 -TargetParameterName "instanceId"
 @{"input_parameter_name1"="input_parameter_value1";
 -MaxConcurrency "a_number_of_instances_or_a_percentage_of_target_set"
 -MaxError "a_number_of_errors_or_a_percentage_of_target_set"
```

The command returns an execution ID. Copy this ID to the clipboard. You can use this ID to view the status of the workflow.
5. Run the following command to view the workflow execution.

Linux

```bash
aws ssm describe-automation-executions \
   --filter Key=ExecutionId,Values="a4a3c0e9-7efd-462a-8594-01234EXAMPLE"
```

Windows

```bash
aws ssm describe-automation-executions ^
   --filter Key=ExecutionId,Values="a4a3c0e9-7efd-462a-8594-01234EXAMPLE"
```

PowerShell

```powershell
Get-SSMAutomationExecutionList | `
   Where {$_AutomationExecutionId -eq "a4a3c0e9-7efd-462a-8594-01234EXAMPLE"}
```

6. To view details about the execution progress, run the following command.

Linux

```bash
aws ssm get-automation-execution \
   --automation-execution-id a4a3c0e9-7efd-462a-8594-01234EXAMPLE
```

Windows

```bash
aws ssm get-automation-execution ^
   --automation-execution-id a4a3c0e9-7efd-462a-8594-01234EXAMPLE
```

PowerShell

```powershell
Get-SSMAutomationExecution `
   -AutomationExecutionId a4a3c0e9-7efd-462a-8594-01234EXAMPLE
```

The system returns information like the following.
Linux

```json
{
  "AutomationExecution": {
    "StepExecutionsTruncated": false,
    "AutomationExecutionStatus": "Success",
    "MaxConcurrency": "1",
    "Parameters": {},
    "MaxErrors": "1",
    "Outputs": {},
    "DocumentName": "AWS-StopEC2Instance",
    "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE",
    "ResolvedTargets": {
      "ParameterValues": [
        "i-02573cafcfEXAMPLE"
      ],
      "Truncated": false
    },
    "ExecutionEndTime": 1564681619.915,
    "Targets": [
      {
        "Values": [
          "DEV"
        ],
        "Key": "tag:ENV"
      }
    ],
    "DocumentVersion": "1",
    "ExecutionStartTime": 1564681576.09,
    "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
    "StepExecutions": [
      {
        "Inputs": {
          "InstanceId": "i-02573cafcfEXAMPLE"
        },
        "Outputs": {},
        "StepName": "i-02573cafcfEXAMPLE",
        "ExecutionEndTime": 1564681619.093,
        "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
        "ExecutionStartTime": 1564681576.836,
        "Action": "aws:executeAutomation",
        "StepStatus": "Success"
      }
    ],
    "TargetParameterName": "InstanceId",
    "Mode": "Auto"
  }
}
```

Windows

```json
{
  "AutomationExecution": {
    "StepExecutionsTruncated": false,
    "AutomationExecutionStatus": "Success",
    "MaxConcurrency": "1",
    "Parameters": {},
    "MaxErrors": "1",
    "Outputs": {},
    "DocumentName": "AWS-StopEC2Instance",
    "AutomationExecutionId": "a4a3c0e9-7efd-462a-8594-01234EXAMPLE",
    "ResolvedTargets": {
      "ParameterValues": [
        "i-02573cafcfEXAMPLE"
      ],
      "Truncated": false
    },
    "ExecutionEndTime": 1564681619.915,
    "Targets": [
      {
        "Values": [
          "DEV"
        ],
        "Key": "tag:ENV"
      }
    ],
    "DocumentVersion": "1",
    "ExecutionStartTime": 1564681576.09,
    "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
    "StepExecutions": [
      {
        "Inputs": {
          "InstanceId": "i-02573cafcfEXAMPLE"
        },
        "Outputs": {},
        "StepName": "i-02573cafcfEXAMPLE",
        "ExecutionEndTime": 1564681619.093,
        "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
        "ExecutionStartTime": 1564681576.836,
        "Action": "aws:executeAutomation",
        "StepStatus": "Success"
      }
    ],
    "TargetParameterName": "InstanceId",
    "Mode": "Auto"
  }
}
```
"i-02573cafcfEXAMPLE"
  },
  "Truncated": false
},
"ExecutionEndTime": 1564681619.915,
"Targets": [
  {
    "Values": [
      "DEV"
    ],
    "Key": "tag:ENV"
  }
],
"DocumentVersion": "1",
"ExecutionStartTimestamp": 1564681576.09,
"ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
"StepExecutions": [
  {
    "Inputs": {
      "InstanceId": "i-02573cafcfEXAMPLE"
    },
    "Outputs": {},
    "StepName": "i-02573cafcfEXAMPLE",
    "ExecutionEndTime": 1564681619.093,
    "StepExecutionId": "86c7b811-3896-4b78-b897-01234EXAMPLE",
    "ExecutionStartTime": 1564681576.836,
    "Action": "aws:executeAutomation",
    "StepStatus": "Success"
  }
],
"TargetParameterName": "InstanceId",
"Mode": "Auto"
}

PowerShell

<table>
<thead>
<tr>
<th>$	extbf{AutomationExecutionId}$</th>
<th>a4a3c0e9-7efd-462a-8594-01234EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$	extbf{AutomationExecutionStatus}$</td>
<td>Success</td>
</tr>
<tr>
<td>$	extbf{CurrentAction}$</td>
<td>:</td>
</tr>
<tr>
<td>$	extbf{CurrentStepName}$</td>
<td>:</td>
</tr>
<tr>
<td>$	extbf{DocumentName}$</td>
<td>AWS-StopEC2Instance</td>
</tr>
<tr>
<td>$	extbf{DocumentVersion}$</td>
<td>: 1</td>
</tr>
<tr>
<td>$	extbf{ExecutedBy}$</td>
<td>arn:aws:sts::123456789012:assumed-role/Administrator/Admin</td>
</tr>
<tr>
<td>$	extbf{ExecutionEndTime}$</td>
<td>8/1/2019 5:46:59 PM</td>
</tr>
<tr>
<td>$	extbf{ExecutionStartTime}$</td>
<td>8/1/2019 5:46:16 PM</td>
</tr>
<tr>
<td>$	extbf{FailureMessage}$</td>
<td>:</td>
</tr>
<tr>
<td>$	extbf{MaxConcurrency}$</td>
<td>: 1</td>
</tr>
<tr>
<td>$	extbf{MaxErrors}$</td>
<td>: 1</td>
</tr>
<tr>
<td>$	extbf{Mode}$</td>
<td>: Auto</td>
</tr>
<tr>
<td>$	extbf{Outputs}$</td>
<td>: {}</td>
</tr>
<tr>
<td>$	extbf{Parameters}$</td>
<td>: {}</td>
</tr>
<tr>
<td>$	extbf{ParentAutomationExecutionId}$</td>
<td>:</td>
</tr>
<tr>
<td>$	extbf{ProgressCounters}$</td>
<td>:</td>
</tr>
<tr>
<td>$	extbf{ResolvedTargets}$</td>
<td>Amazon.SimpleSystemsManagement.Model.ResolvedTargets</td>
</tr>
<tr>
<td>$	extbf{StepExecutions}$</td>
<td>: (i-02573cafcfEXAMPLE)</td>
</tr>
<tr>
<td>$	extbf{StepExecutionsTruncated}$</td>
<td>: False</td>
</tr>
<tr>
<td>$	extbf{Target}$</td>
<td>:</td>
</tr>
<tr>
<td>$	extbf{TargetLocations}$</td>
<td>: {}</td>
</tr>
<tr>
<td>$	extbf{TargetMaps}$</td>
<td>: {}</td>
</tr>
<tr>
<td>$	extbf{TargetParameterName}$</td>
<td>: InstanceId</td>
</tr>
<tr>
<td>$	extbf{Targets}$</td>
<td>: (tag:Name)</td>
</tr>
</tbody>
</table>
Working with Automation executions

**Note**
You can also monitor the status of the workflow in the console. In the execution list, choose the execution you just ran and then choose the **Steps** tab. This tab shows the status of the workflow actions.

**About targets**

The **Targets** parameter enables you to quickly define which resources in your fleet can run an Automation workflow. For example, if you want to run an Automation that restarts your managed instances, then instead of manually selecting dozens of instance IDs in the console or typing them in a command, you can target instances by specifying EC2 tags with the **Targets** parameter.

When you run an Automation that uses a target, Systems Manager creates a child Automation for each target. For example, if you target Amazon Elastic Block Store (Amazon EBS) volumes by specifying tags, and those tags resolve to 100 Amazon EBS volumes, then Systems Manager creates 100 child Automation workflows. The parent Automation is complete when all child Automations reach a final state.

**Note**
Any **input parameters** that you specify at runtime (either in the **Input parameters** section of the console or by using the **parameters** option from the command line) are automatically processed by all child Automations.

You can target resources for an Automation execution by using tags, Resource Groups, and parameter values. Additionally, you can use the **TargetMaps** option to target multiple parameter values from the command line or a file. The following section describes each of these targeting options in more detail.

**Targeting tags**

Many AWS resources support tags, including Amazon EC2 and Amazon Relational Database Service (Amazon RDS) instances, Amazon Elastic Block Store (Amazon EBS) volumes and snapshots, Resource Groups, and Amazon Simple Storage Service (Amazon S3) buckets, to name a few. You can quickly run Automation workflows on your AWS resources by targeting tags. A tag is a key-value pair, such as Operating_System-Linux or Department-Finance. If you assign a specific name to a resource, then you can also use the word "Name" as a key, and the name of the resource as the value.

When you specify a tag as the target for an Automation, you also specify a target parameter. The target parameter uses the **TargetParameterName** option. By choosing a target parameter, you define the type of resource on which the Automation runs. The target parameter you specify with the tag must be a valid parameter defined in the Automation document. For example, if you want to target dozens of EC2 instances by using tags, then choose the **InstanceId** target parameter. By choosing this parameter, you define instances as the resource type for the Automation execution. The following screenshot uses the AWS-DetachEBSVolume document. The logical target parameter is **VolumeId**.
The AWS-DetachEBSVolume document also includes a special property called **Target type**, which is set to /AWS::EC2::Volume. This means that if the tag-key pair Finance-TestEnv returns different types of resources (for example, EC2 instances, Amazon EBS volumes, Amazon EBS snapshots) then only Amazon EBS volumes will be used.

**Important**
Target parameter names are case sensitive. If you run Automations by using either the AWS CLI or AWS Tools for Windows PowerShell, then you must enter the target parameter name exactly as it is defined in the Automation document. If you don’t, the system returns an InvalidAutomationExecutionParametersException error. You can use the DescribeDocument API action to see information about the available target parameters in a specific document. Here is an example AWS CLI command that provides information about the AWS-DeleteSnapshot document:

```
aws ssm describe-document --name AWS-DeleteSnapshot
```

Here are some example AWS CLI commands that target resources by using tags.

**Example 1: Targeting tags using a key-value pair to restart EC2 instances**

This example restarts all EC2 instances that are tagged with a key of Department and a value of HumanResources. The target parameter uses the InstanceId parameter from the Automation document. The example uses an additional parameter to run the automation by using an Automation service role (also called an assume role).

```
aws ssm start-automation-execution --document-name AWS-RestartEC2Instance --targets
Key=tag:Department,Values=HumanResources --target-parameter-name InstanceId --parameters
"AutomationAssumeRole=arn:aws:iam::111122223333:role/AutomationServiceRole"
```

**Example 2: Targeting tags using a key-value pair to delete Amazon EBS snapshots**

The following example uses the AWS-DeleteSnapshot Automation document to delete all snapshots with a key of Name and a value of January2018Backups. The target parameter uses the Volumeld parameter.
aws ssm start-automation-execution --document-name AWS-DeleteSnapshot --targets Key=tag:Name,Values=January2018Backups --target-parameter-name VolumeId

Targeting AWS Resource Groups

You can specify a single AWS resource group as the target of an Automation. Systems Manager creates a child Automation for every object in the target Resource Group.

For example, say that one of your Resource Groups is named PatchedAMIs. This Resource Group includes a list of 25 Windows Amazon Machine Images (AMIs) that are routinely patched. If you run an Automation that uses the AWS-CreateManagedWindowsInstance document and target this Resource Group, then Systems Manager creates a child Automation for each of the 25 AMIs. This means, that by targeting the PatchedAMIs Resource Group, the Automation creates 25 instances from a list of patched AMIs. The parent Automation is complete when all child Automations complete processing or reach a final state.

The following AWS CLI command applies to the PatchAMIs Resource Group example. The command takes the AmiId parameter for the --target-parameter-name option. The command doesn't include an additional parameter defining which type of instance to create from each AMI. The AWS-CreateManagedWindowsInstance document defaults to the t2.medium instance type, so this command would create 25 t2.medium EC2 instances for Windows Server.

aws ssm start-automation-execution --document-name AWS-CreateManagedWindowsInstance --targets Key=ResourceGroup,Values=PatchedAMIs --target-parameter-name AmiId

The following console example uses a Resource Group called t2-micro-instances.

<table>
<thead>
<tr>
<th>Targets and Rate Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable the targets selection and rate control. Available for Execution mode &quot;Auto&quot; only.</td>
</tr>
<tr>
<td>Enable targets and rate control</td>
</tr>
<tr>
<td>Targets</td>
</tr>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>Resource group</td>
</tr>
</tbody>
</table>

Targeting parameter values

You can also target a parameter value. You enter ParameterValues as the key and then enter the specific resource value where you want the Automation workflow to run. If you specify multiple values, Systems Manager runs a child Automation workflow on each value specified.

For example, say that your Automation document includes an InstanceID parameter. If you target the values of the InstanceID parameter when you run the Automation, then Systems Manager runs a child Automation for each instance ID value specified. The parent Automation is complete when the Automation finishes running each specified instance, or if the execution fails. You can target a maximum of 50 parameter values.
The following example uses the AWS-CreateImage Automation document. The target parameter name specified is `InstanceId`. The key uses `ParameterValues`. The values are two EC2 instance IDs. This command creates an Automation workflow for each instance, which produces an AMI from each instance.

```bash
aws ssm start-automation-execution --document-name AWS-CreateImage --target-parameter-name InstanceId --targets Key=ParameterValues,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE
```

**Note**

AutomationAssumeRole is not a valid parameter. Don't choose this item when running Automation workflows that target a parameter value.

**Targeting parameter value maps**

The `TargetMaps` option expands your ability to target `ParameterValues`. You can enter an array of parameter values by using `TargetMaps` at the command line. You can specify a maximum of 50 parameter values at the command line. If you want to run commands that specify more than 50 parameter values, then you can enter the values in a JSON file. You can then call the file from the command line.

**Note**

`TargetMaps` are not supported in the console.

Use the following format to specify multiple parameter values by using the `TargetMaps` option in a command:

```bash
```

If you want to enter more than 50 parameter values for the `TargetMaps` option, then specify the values in a file by using the following JSON format. Using a JSON file also improves readability when providing multiple parameter values.

```json
[
    {
        "parameterA": "parameterValueA1", 
        "parameterB": "parameterValueB1", 
        "parameterC": "parameterValueC1"
    },
    {
        "parameterA": "parameterValueA2", 
        "parameterB": "parameterValueB2", 
        "parameterC": "parameterValueC2"
    },
    {
        "parameterA": "parameterValueA3", 
        "parameterB": "parameterValueB3", 
        "parameterC": "parameterValueC3"
    }
]
```

Save the file with a `.json` file extension. You can call the file by using the following command:

```bash
aws ssm start-automation-execution --document-name name_of_document --parameters one_or_more_input_parameters --target-maps full_path_to_file/file_name.json
```

You can also download the file from an S3 bucket, as long as you have permission to read data from the bucket. Use the following command format:

```bash
```
Here is an example scenario to help you understand the TargetMaps option. In this scenario, a user wants to create EC2 instances of different types from different AMIs. To perform this task, the user creates an Automation document named AMI_Testing. This document defines two input parameters: instanceType and imageId.

```json
{
  "description": "AMI Testing",
  "schemaVersion": "0.3",
  "assumeRole": "{{assumeRole}}",
  "parameters": {
    "assumeRole": {
      "type": "String",
      "description": "Role under which to run the automation",
      "default": ""
    },
    "instanceType": {
      "type": "String",
      "description": "Type of EC2 Instance to launch for this test"
    },
    "imageId": {
      "type": "String",
      "description": "Source AMI id from which to run instance"
    }
  },
  "mainSteps": [
    {
      "name": "runInstances",
      "action": "aws:runInstances",
      "maxAttempts": 1,
      "onFailure": "Abort",
      "inputs": {
        "ImageId": "{{imageId}}",
        "InstanceType": "{{instanceType}}",
        "MinInstanceCount": 1,
        "MaxInstanceCount": 1
      }
    }
  ],
  "outputs": [
    "runInstances.InstanceIds"
  ]
}
```

The user then specifies the following target parameter values in a file named AMI_instance_types.json.

```json
[
  {
    "instanceType" : ["t2.micro"],
    "imageId" : ["ami-b70554c8"]
  },
  {
    "instanceType" : ["t2.small"],
    "imageId" : ["ami-b70554c8"]
  },
  {
    "instanceType" : ["t2.medium"],
    "imageId" : ["ami-cfe4b2b0"]
  },
  {
    "instanceType" : ["t2.medium"],
    "imageId" : ["ami-cfe4b2b0"]
  },
  {
    "instanceType" : ["t2.medium"],
    "imageId" : ["ami-cfe4b2b0"]
  }
]```
The user can run the Automation and create the five EC2 instances defined in AMI_instance_types.json by running the following command:

```
aws ssm start-automation-execution --document-name AMI_Testing --target-parameter-name imageId --target-maps file:///home/TestUser/workspace/runinstances/AMI_instance_types.json
```

**Targeting all instances**

You can run an Automation on all managed instances in the current AWS account and Region by choosing **All instances** in the **Targets** list. For example, if you want to restart all managed instances your AWS account and the current Region, you can choose the **AWS-RestartEC2Instance** playbook and then choose **All instances** from the **Targets** list.

After you choose **All instances**, Systems Manager populates the **Instance** field with an asterisk (*) and makes the field unavailable for changes (the field is grayed out). Systems Manager also makes the **InstanceId** field in the **Input parameters** field unavailable for changes. Making these fields unavailable for changes is expected behavior if you choose to target all instances.

**About concurrency and error thresholds**

You can control the execution of an Automation workflow across a fleet of AWS resources by specifying a concurrency value and an error threshold. Concurrency and error threshold are collectively called **rate controls**.

**Concurrency**

Concurrency enables you to specify how many resources are allowed to run an Automation simultaneously. Concurrency helps to limit the impact or downtime on your resources when processing an Automation. You can specify either an absolute number of resources, for example 20, or a percentage of the target set, for example 10%.

The queueing system delivers the Automation to a single resource and waits until the initial invocation is complete before sending the Automation to two more resources. The system exponentially sends the Automation to more resources until the concurrency value is met.

**Error Thresholds**

An error threshold enables you to specify how many Automation workflows are allowed to fail before Systems Manager stops sending the Automation to other resources. You can specify either an absolute number of errors, for example 10, or a percentage of the target set, for example 10%.
If you specify an absolute number of 3 errors, for example, the system stops sending the Automation when the third error is received. If you specify 1, then the system stops sending the Automation to additional resources after the first error result is returned.

If you send an Automation to, for example, 50 instances and set the error threshold to 10%, then the system stops sending the command to additional instances when the fifth error is received. Invocations that are already running an Automation when an error threshold is reached are allowed to be completed, but some of these Automations might fail as well. If you need to ensure that there won’t be more errors than the number specified for the error threshold, then set the Concurrency value to 1 so that Automations proceed one at a time.

Running Automation workflows based on triggers

This section includes information about how to run Automation workflows using a trigger. Automation workflows can be initiated by several different triggers, such as Amazon CloudWatch Events, State Manager Associations, or maintenance windows. By using triggers, you can run Automation workflows as a result of a specific event or on a scheduled basis.

Contents
- Running Automation workflows with triggers using CloudWatch Events (p. 314)
- Running Automation workflows with triggers using State Manager (p. 319)
- Running Automation workflows with triggers using Maintenance Windows (p. 327)

Running Automation workflows with triggers using CloudWatch Events

You can start an Automation workflow by specifying an Automation document as the target of an Amazon CloudWatch event. You can start workflows according to a schedule, or when a specific AWS system event occurs. For example, let’s say you create an Automation document named BootStrapInstances that installs software on an instance when an instance starts. To specify the BootStrapInstances document (and corresponding workflow) as the target of a CloudWatch event, you first create a new CloudWatch Events rule. (Here’s an example rule: Service name: EC2, Event Type: EC2 Instance State-change Notification, Specific state(s): running, Any instance.) Then you use the following procedures to specify the BootStrapInstances document as the target of the event using the CloudWatch console, AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell. When a new instance starts, the system runs the workflow and installs software.

For information about creating Automation documents, see Working with Automation documents (Playbooks) (p. 353).

Creating a CloudWatch event that runs an Automation workflow (console)

Use the following procedure to configure an Automation workflow as the target of a CloudWatch event.

To configure an Automation document as a target of a CloudWatch event rule

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Choose Event Pattern or Schedule. Event Pattern lets you build a rule that generates events for specific actions in AWS services. Schedule lets you build a rule that generates events according to a schedule that you specify by using the cron format.
4. Choose the remaining options for the rule you want to create, and then choose Add target.
5. In the Select target type list, choose SSM Automation.
6. In the Document list, choose an Automation document to run when your target is invoked.
7. Expand **Configure document version**, and choose a version. $DEFAULT was explicitly set as the default document version in Systems Manager. You can choose a specific version, or use the latest version.

8. Expand **Configure automation parameter(s)**, and either keep the default parameter values (if available) or enter your own values.

   **Note**
   Required parameters have an asterisk (*) next to the parameter name. To create a target, you must specify a value for each required parameter. If you don’t, the system creates the rule, but it won’t run.

9. In the permissions section, choose an option. CloudWatch uses the role to start the Automation workflow.

10. Choose **Configure details** and complete the wizard.

**Create a CloudWatch event that runs an Automation document (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to create a CloudWatch event rule and configure an Automation document as the target.

**To configure an Automation document as a target of a CloudWatch event rule**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Create a command to specify a new CloudWatch event rule. Here are some template commands to help.

   **Triggers based on a schedule**

   **Linux**

   ```bash
   aws events put-rule \ 
   --name "rule_name" \ 
   --schedule-expression "cron_or_rate_expression"
   ```

   **Windows**

   ```bash
   aws events put-rule ^
   --name "rule_name" ^
   --schedule-expression "cron_or_rate_expression"
   ```

   **PowerShell**

   ```powershell
   Write-CWERule ` 
   -Name "rule_name" ` 
   -ScheduleExpression "cron_or_rate_expression"
   ```

   The following example creates a CloudWatch event rule that triggers every day at 9:00am (UTC).

   **Linux**

   ```bash
   aws events put-rule \ 
   --name "DailyAutomationRule" \ 
   --schedule-expression "cron(0 9 * * *)"
   ```
Working with Automation executions

**Windows**

```bash
aws events put-rule ^
   --name "DailyAutomationRule" ^
   --schedule-expression "cron(0 9 * * ? *)"
```

**PowerShell**

```powershell
Write-CWERule ` `-Name "DailyAutomationRule` `-ScheduleExpression "cron(0 9 * * ? *)"
```

**Triggers based on an event**

**Linux**

```bash
aws events put-rule \  
   --name "rule_name" \  
   --event-pattern "{\"source\":[\"aws.service\"],\"detail-type\": 
   [\"service_event_detail_type\"]}"
```

**Windows**

```bash
aws events put-rule ^
   --name "rule_name" ^
   --event-pattern "{\"source\":[\"aws.service\"],\"detail-type\": 
   [\"service_event_detail_type\"]}"
```

**PowerShell**

```powershell
Write-CWERule ` `-Name "rule_name` `-EventPattern '{"source":["aws.service"],"detail-type": 
   ["service_event_detail_type"]]'
```

The following example creates a CloudWatch event rule that triggers when any EC2 instance in the Region changes state.

**Linux**

```bash
aws events put-rule \  
   --name "EC2InstanceStateChanges" \  
   --event-pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
```

**Windows**

```bash
aws events put-rule ^
   --name "EC2InstanceStateChanges" ^
   --event-pattern "{\"source\":[\"aws.ec2\"],\"detail-type\":[\"EC2 Instance State-change Notification\"]}"
```
The command returns details for the new CloudWatch rule similar to the following.

Linux

```json
{
  "RuleArn": "arn:aws:events:us-east-1:123456789012:rule/automationrule"
}
```

Windows

```json
{
  "RuleArn": "arn:aws:events:us-east-1:123456789012:rule/automationrule"
}
```

PowerShell

```powershell
Write-CWERule `-Name "EC2InstanceStateChanges" `-EventPattern '{"source": ["aws.ec2"], "detail-type": ["EC2 Instance State-change Notification"]}
```

3. Create a command to specify an Automation document as a target of the CloudWatch event rule you created in step 2. Here are some template commands to help.

Linux

```bash
aws events put-targets
  --rule CW_Event_Rule_Name
  --targets '({"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-definition/Automation_Document_Name","Input":{"DocumentParameter": [{"ParameterValue":},{"AutomationAssumeRole": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]},"Id": "Target_Id","RoleArn": "arn:aws:iam::123456789012:role/service-role/CWE_Role_Name_To_Run_Automation"})'
```

Windows

```bash
aws events put-targets
  --rule CW_Event_Rule_Name
  --targets '({"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-definition/Automation_Document_Name","Input":{"DocumentParameter": [{"ParameterValue":},{"AutomationAssumeRole": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]},"Id": "Target_Id","RoleArn": "arn:aws:iam::123456789012:role/service-role/CWE_Role_Name_To_Run_Automation"})'
```

PowerShell

```powershell
$Target = New-Object Amazon.CloudWatchEvents.Model.Target
$Target.Id = "Target_Id"
```
The following example creates a CloudWatch event target that starts the specified instance ID using the document `AWS-StartEC2Instance`.

Linux

```bash
aws events put-targets \
  --rule DailyAutomationRule \
  --targets '{"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-definition/AWS-StartEC2Instance","Input": "{\"InstanceId\": ["i-02573cafcfEXAMPLE"], \"AutomationAssumeRole\": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]},\"Id": "Target1","RoleArn": "arn:aws:iam::123456789012:role/service-role/AWS_Events_Invoke_Start_Automation_Execution_1213609520\"'}
```

Windows

```bash
aws events put-targets ^
  --rule DailyAutomationRule ^
  --targets '{"Arn": "arn:aws:ssm:us-east-1:123456789012:automation-definition/AWS-StartEC2Instance","Input": "{\"InstanceId\": ["i-02573cafcfEXAMPLE"], \"AutomationAssumeRole\": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]},\"Id": "Target1","RoleArn": "arn:aws:iam::123456789012:role/service-role/AWS_Events_Invoke_Start_Automation_Execution_1213609520\"'}
```

PowerShell

```powershell
$Target = New-Object Amazon.CloudWatchEvents.Model.Target
$Target.Id = "Target1"
$Target.RoleArn = "arn:aws:iam::123456789012:role/service-role/AWS_Events_Invoke_Start_Automation_Execution_1213609520"
$Target.Input = '{"InstanceId": ["i-02573cafcfEXAMPLE"], "AutomationAssumeRole": ["arn:aws:iam::123456789012:role/AutomationServiceRole"]}'
Write-CWETarget \
  -Rule "DailyAutomationRule" \
  -Target $Target
```

The system returns information like the following.

Linux

```json
{
  "FailedEntries": [],
  "FailedEntryCount": 0
}
```
Windows

```
{
    "FailedEntries": [],
    "FailedEntryCount": 0
}
```

PowerShell

There is no output if the command succeeds for PowerShell.

Running Automation workflows with triggers using State Manager

You can start an Automation workflow by creating a State Manager association with an Automation document. By creating a State Manager association with an Automation document, you can target different types of AWS resources. For example, you can create associations that enforce a desired state on an AWS resource, including the following:

- Attach a Systems Manager role to EC2 instances to make them managed instances.
- Enforce desired ingress and egress rules for a security group.
- Create or delete Amazon DynamoDB (DynamoDB) backups.
- Create or delete Amazon Elastic Block Store (Amazon EBS) snapshots.
- Disable read and write permissions on Amazon Simple Storage Service (Amazon S3) buckets.
- Start, restart, or stop managed instances and Amazon Relational Database Service (Amazon RDS) instances.
- Patch Windows and Linux AMIs.

Use the following procedures to create a State Manager Association that runs an Automation workflow using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell.

Before You Begin

Be aware of the following important details before you run Automation workflows by using State Manager.

- Before you can create an association that runs an Automation document, verify that you configured permissions for Systems Manager Automation. For more information, see Getting started with Automation (p. 281).
- State Manager associations that run Automation documents contribute to the maximum number of concurrently running Automations in your AWS account. You can have a maximum of 25 concurrent Automations running with a maximum of 75 child Automations running at one time. For information, see Systems Manager service quotas in the Amazon Web Services General Reference.
- Systems Manager automatically creates a service-linked role so that State Manager has permission to call Systems Manager Automation API actions. If you want, you can create the service-linked role yourself by running the following command from the AWS CLI or AWS Tools for PowerShell.

```
aws iam create-service-linked-role \
--aws-service-name ssm.amazonaws.com
```
For more information about service-linked roles, see Using service-linked roles for Systems Manager (p. 1122).

Creating an association that runs an Automation workflow (console)

The following procedure describes how to use the Systems Manager console to create a State Manager association that runs an Automation workflow.

To create a State Manager association that runs a Systems Manager Automation workflow

2. In the navigation pane, choose State Manager, and then choose Create association.
3. In the Name field, specify a name. This is optional, but recommended.
4. In the Document list, choose a document. Use the Search bar to filter on Document type : Equal : Automation documents. To view more Automation documents, use the numbers to the right of the Search bar.

   **Note**
   You can view information about a document by choosing the document name.

5. Choose Simple execution to run the automation on one or more targets by specifying the resource ID for those targets. Choose Rate control to run the automation across a fleet of AWS resources by specifying a targeting option such as tags or AWS Resource Groups. You can also control the execution of the automation across your resources by specifying concurrency and error thresholds.

   **Note**
   If you chose Rate control, the Targets section appears.

6. In the Targets section, choose a method for targeting resources.

   a. (Required) In the Parameter list, choose a parameter. The items in the Parameter list are determined by the parameters in the Automation document that you selected at the start of this procedure. By choosing a parameter, you define the type of resource on which the Automation workflow runs.

   b. (Required) In the Targets list, choose a method for targeting the resources.

      - **Resource Group**: Choose the name of the group from the Resource Group list. For more information about targeting AWS Resource Groups in Automation documents, see Targeting AWS Resource Groups (p. 310).
      - **Tags**: Enter the tag key and (optionally) the tag value in the fields provided. Choose Add. For more information about targeting tags in Automation documents, see Targeting tags (p. 308).
      - **Parameter Values**: Enter values in the Input parameters section. If you specify multiple values, Systems Manager runs a child Automation workflow on each value specified.

      For example, say that your Automation document includes an InstanceID parameter. If you target the values of the InstanceID parameter when you run the Automation, then
Systems Manager runs a child Automation for each instance ID value specified. The parent Automation is complete when the Automation finishes running each specified instance, or if the execution fails. You can target a maximum of 50 parameter values. For more information about targeting parameter values in Automation documents, see Targeting parameter values (p. 310).

7. In the **Input parameters** section, specify the required input parameters.

   If you chose to target resources by using tags or a resource group, then you may not need to choose some of the options in the Input parameters section. For example, if you chose the AWS-RestartEC2Instance document, and you chose to target instances by using tags, then you don't need to specify or choose instance IDs in the Input parameters section. The Automation execution locates the instances to restart by using the tags you specified.

   **Important**
   You must specify a role ARN in the AutomationAssumeRole field. State Manager uses the assume role to call AWS services specified in the Automation document and run Automation associations on your behalf. For more information, see Running an Automation workflow by using an IAM service role (p. 337).

8. In the **Specify schedule** section, choose On Schedule if you want to run the association at regular intervals. If you choose this option, then use the options provided to create the schedule using Cron or Rate expressions. For more information about Cron and Rate expressions for State Manager, see Cron and rate expressions for associations (p. 1193).

   **Note**
   Rate expressions are the preferred scheduling mechanism for State Manager associations that run Automation documents. Rate expressions allow more flexibility for running associations in the event that you reach the maximum number of concurrently running Automations. With a rate schedule, Systems Manager can retry the Automation shortly after receiving notification that concurrent Automations have reached their maximum and have been throttled.

   Choose **No schedule** if you want to run the association one time.

9. (Optional) In the **Rate Control** section, choose **Concurrency** and **Error threshold** options to control the Automation execution across your AWS resources.

   a. In the **Concurrency** section, choose an option:

      - Choose **targets** to enter an absolute number of targets that can run the Automation workflow simultaneously.
      - Choose **percentage** to enter a percentage of the target set that can run the Automation workflow simultaneously.

   b. In the **Error threshold** section, choose an option:

      - Choose **errors** to enter an absolute number of errors allowed before Automation stops sending the workflow to other resources.
      - Choose **percentage** to enter a percentage of errors allowed before Automation stops sending the workflow to other resources.

   For more information about using targets and rate controls with Automation, see Running Automation workflows that use targets and rate controls (p. 299).

10. Choose **Create Association**.

    **Important**
    When you create an association, the association immediately runs against the specified targets. The association then runs based on the cron or rate expression you chose. If you chose **No schedule**, the association does not run again.
Creating an association that runs an Automation workflow (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to create a State Manager association that runs an Automation workflow.

**To create an association that runs an Automation workflow**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.
   
   For information, see [Install or upgrade AWS command line tools](#).

2. Run the following command to view a list of documents.
   
   **Linux**
   
   ```bash
   aws ssm list-documents
   ```

   **Windows**
   
   ```bash
   aws ssm list-documents
   ```

   **PowerShell**
   
   ```powershell
   Get-SSMDocumentList
   ```

   Note the name of the Automation document that you want to use for the association.

3. Run the following command to view details about the Automation document.
   
   **Linux**
   
   ```bash
   aws ssm describe-document \
   --name document_name
   ```

   Note a parameter name (for example, `InstanceId`) that you want to use for the `--automation-target-parameter-name` option. This parameter determines the type of resource on which the Automation runs.

   **Windows**
   
   ```bash
   aws ssm describe-document ^
   --name document_name
   ```

   Note a parameter name (for example, `InstanceId`) that you want to use for the `--automation-target-parameter-name` option. This parameter determines the type of resource on which the Automation runs.

   **PowerShell**
   
   ```powershell
   Get-SSMDocumentDescription `^`
   -Name document_name
   ```

   Note a parameter name (for example, `InstanceId`) that you want to use for the `AutomationTargetParameterName` option. This parameter determines the type of resource on which the Automation runs.

4. Create a command that runs an Automation workflow using a State Manager association. Here are some template commands to help.
Targeting using tags

Linux

```bash
aws ssm create-association \
  --association-name AssociationName \
  --targets Key=tag:TagKey,Values=TagValue \
  --name AutomationDocumentName \
  --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM,(Additional parameters, if any) \
  --automation-target-parameter-name (parameter to target) \
  --schedule "cron_or_rate_expression"
```

**Note**
If you create an association by using the AWS CLI, use the `--targets` parameter to target instances for the association. Don't use the `--instance-id` parameter. The `--instance-id` parameter is a legacy parameter.

Windows

```bash
aws ssm create-association ^
  --association-name AssociationName ^
  --targets Key=tag:TagKey,Values=TagValue ^
  --name AutomationDocumentName ^
  --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM,(Additional parameters, if any) ^
  --automation-target-parameter-name (parameter to target) ^
  --schedule "cron_or_rate_expression"
```

**Note**
If you create an association by using the AWS CLI, use the `--targets` parameter to target instances for the association. Don't use the `--instance-id` parameter. The `--instance-id` parameter is a legacy parameter.

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag:TagKey"
$Targets.Values = "TagValue"

New-SSMAssociation \
  -AssociationName "AssociationName" \
  -Target $Targets \
  -Name "AutomationDocumentName" \
  -Parameters @{
    "AutomationAssumeRole"="arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM; (Additional parameters, if any)" 
  } \
  -AutomationTargetParameterName "parameter_to_target" \
  -ScheduleExpression "cron_or_rate_expression"
```

**Note**
If you create an association by using the AWS Tools for PowerShell, use the `Target` parameter to target instances for the association. Don't use the `InstanceId` parameter. The `InstanceId` parameter is a legacy parameter.

Targeting using parameter values
Linux

```bash
aws ssm create-association
--association-name AssociationName
--targets Key=ParameterValues,Values=value_1,value_2,value_3
--name AutomationDocumentName
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM,(Additional parameters, if any)
--automation-target-parameter-name (parameter to target)
--schedule "cron_or_rate_expression"
```

Windows

```bash
aws ssm create-association
--association-name AssociationName
--targets Key=ParameterValues,Values=value_1,value_2,value_3
--name AutomationDocumentName
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM,(Additional parameters, if any)
--automation-target-parameter-name (parameter to target)
--schedule "cron_or_rate_expression"
```

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ParameterValues"
$Targets.Values = "value_1","value_2","value_3"

New-SSMAssociation
-AssociationName "AssociationName"
-Target $Targets
-Name "AutomationDocumentName"
-Parameters @{
  "AutomationAssumeRole"="arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM;
  (Additional parameters, if any)"
  -AutomationTargetParameterName "parameter_to_target"
  -ScheduleExpression "cron_or_rate_expression"
}
```

Targeting using AWS Resource Groups

Linux

```bash
aws ssm create-association
--association-name AssociationName
--targets Key=ResourceGroup,Values=Resource_Group_name
--name AutomationDocumentName
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM,(Additional parameters, if any)
--automation-target-parameter-name (parameter to target)
--schedule "cron_or_rate_expression"
```

Windows

```bash
aws ssm create-association
--association-name AssociationName
--targets Key=ResourceGroup,Values=Resource_Group_name
--name AutomationDocumentName
```
---parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM,(Additional parameters, if any) ^
--automation-target-parameter-name (parameter to target) ^
--schedule "cron_or_rate_expression"

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "Resource_Group_Name"

New-SSMAssociation
-AssociationName "AssociationName" 
-Target $Targets 
-Name "AutomationDocumentName" 
-Parameters {
  "AutomationAssumeRole"="arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM; (Additional parameters, if any)" 
-AutomationTargetParameterName "parameter_to_target" 
-ScheduleExpression "cron_or_rate_expression"
}

The command returns details for the new association similar to the following.

Linux

```json
{
  "AssociationDescription": {
    "ScheduleExpression": "cron(0 7 ? * MON *)",
    "Name": "AWS-StartEC2Instance",
    "Parameters": {
      "AutomationAssumeRole": ["arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM"
    ]
  },
  "Overview": {
    "Status": "Pending",
    "DetailedStatus": "Creating"
  },
  "AssociationId": "1450b4b7-bea2-4e4b-b340-01234EXAMPLE",
  "DocumentVersion": "$DEFAULT",
  "AutomationTargetParameterName": "InstanceId",
  "LastUpdateAssociationDate": 1564686638.498,
  "Date": 1564686638.498,
  "AssociationVersion": "1",
  "AssociationName": "CLI",
  "Targets": [
    {
      "Values": [
        "DEV"
      ],
      "Key": "tag:ENV"
    }
  ]
}
```

Windows

```json
{
```
"AssociationDescription": {
    "ScheduleExpression": "cron(0 7 ? * MON *)",
    "Name": "AWS-StartEC2Instance",
    "Parameters": {
        "AutomationAssumeRole": [
            "arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM"
        ],
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "1450b4b7-bea2-4e4b-b340-01234EXAMPLE",
        "DocumentVersion": "$DEFAULT",
        "AutomationTargetParameterName": "InstanceId",
        "LastUpdateAssociationDate": 1564686638.498,
        "Date": 1564686638.498,
        "AssociationVersion": "1",
        "AssociationName": "CLI",
        "Targets": [
            {
                "Values": [
                    "DEV"
                ],
                "Key": "tag:ENV"
            }
        ]
    }
}

PowerShell

<table>
<thead>
<tr>
<th>Name</th>
<th>AWS-StartEC2Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceId</td>
<td>:</td>
</tr>
<tr>
<td>Date</td>
<td>8/1/2019 7:31:38 PM</td>
</tr>
<tr>
<td>Status.Name</td>
<td>:</td>
</tr>
<tr>
<td>Status.Message</td>
<td>:</td>
</tr>
<tr>
<td>Status.Date</td>
<td>:</td>
</tr>
<tr>
<td>Status.AdditionalInfo</td>
<td>:</td>
</tr>
</tbody>
</table>

**Note**

If you use tags to create an association on one or more target instances, and then you remove the tags from an instance, that instance no longer runs the association. The instance is disassociated from the State Manager document.

**Troubleshooting State Manager Automation executions**

Systems Manager Automation enforces a limit of 25 concurrent executions, 75 child executions, and 1,000 queued executions per account, per Region. If a State Manager association that runs an Automation document shows a status of **Failed** and a detailed status of **AutomationExecutionLimitExceeded**, then your execution may have reached the limit. As a result, Systems Manager throttles the executions. To resolve this issue, do the following:

- Use a different rate or cron expression for your association. For example, if the association is scheduled to run every 30 minutes, then change the expression so that it runs every hour or two.
- Delete existing Automation executions that have a status of **Pending**. By deleting these executions, you clear the current queue.
Running Automation workflows with triggers using Maintenance Windows

You can start an Automation workflow by configuring an Automation document as a registered task for a maintenance window. By registering the Automation document as a registered task, the maintenance window runs the automation workflow during the scheduled maintenance period.

For example, let's say you create an Automation document named `CreateAMI` that creates an Amazon Machine Image (AMI) of instances registered as targets to the maintenance window. To specify the `CreateAMI` document (and corresponding workflow) as a registered task of a maintenance window, you first create a maintenance window and register targets. Then you use the following procedure to specify the `CreateAMI` document as a registered task within the maintenance window. When the maintenance window starts during the scheduled period, the system runs the automation workflow and creates an AMI of the registered targets.

For information about creating Automation documents, see Working with Automation documents (Playbooks) (p. 353).

Use the following procedures to configure an Automation workflow as a registered task for a maintenance window using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), or AWS Tools for Windows PowerShell.

Registering an Automation workflow task to a maintenance window (console)

The following procedure describes how to use the Systems Manager console to configure an Automation workflow as a registered task for a maintenance window.

**Before You Begin**

Before you complete the following procedure, you must create a maintenance window and register at least one target. For more information, see the following procedures:

- Create a maintenance window (console) (p. 634).
- Assign targets to a maintenance window (console) (p. 635)

**To configure an Automation workflow as a registered task for a maintenance window**

2. In the left navigation pane, choose Maintenance Windows, and then choose the maintenance window you want to register an Automation task with.
3. Choose Actions. Then choose Register Automation task to run your choice of an Automation workflow on targets by using an Automation document.
4. For Name, enter a name for the task.
5. For Description, enter a description.
6. For Document, choose the Automation document that defines the tasks to run.
7. For Document version, choose the document version to use.
8. For Task priority, specify a priority for this task. 1 is the highest priority. Tasks in a maintenance window are scheduled in priority order; tasks that have the same priority are scheduled in parallel.
9. In the Targets section, identify the targets on which you want to run this automation workflow by specifying tags or by selecting instances manually.

**Important**

If you choose an Automation document that doesn't target managed instances, you must still select at least one maintenance window target. In this situation, we recommend registering a target for a tag key-value pair that is not used by your managed instances. For example, if you choose the Automation document `AWS-CopySnapshot`, then the resulting automation workflow targets Amazon Elastic Block Store (EBS) snapshots instead of managed instances. In this case, you can register a target to your maintenance window,
which targets a tag key-value pair that is not used by your managed instances, such as key=MaintenanceWindow and value=Snapshot.

10. (Optional) For Rate control:

   • For Concurrency, specify either a number or a percentage of targets on which to run the automation workflow at the same time.

     Note
     If you selected targets by choosing tag key-value pairs, and you are not certain how many targets use the selected tags, then limit the number of automation workflows that can run at the same time by specifying a percentage.

     When the maintenance window runs, a new Automation execution is initiated per target. There is a limit of 25 concurrent executions of Automation and 75 child executions of Automation per AWS account. If you specify a concurrency rate greater than 25, concurrent executions greater than 25 are automatically added to the execution queue.

     For information, see Systems Manager service quotas in the Amazon Web Services General Reference.

   • For Error threshold, specify when to stop running the automation workflow on other targets after it fails on either a number or a percentage of targets. For example, if you specify three errors, then Systems Manager stops running automation workflows when the fourth error is received. Targets still processing the workflow might also send errors.

11. In the IAM service role area, choose one of the following options to provide permissions for Systems Manager to start the Automation workflow:

   • Create and use a service-linked role for Systems Manager

     Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role. Additionally, AWS automatically defines and sets the permissions of service-linked roles, depending on the actions that the linked service performs on your behalf.

     Note
     If a service-linked role has already been created for your account, choose Use the service-linked role for Systems Manager.

   • Use a custom service role

     If you want to use stricter permissions than those provided by the service-linked role, you can create a custom service role for maintenance window tasks.

     To create a custom service role, see one of the following topics:

     • Control access to maintenance windows (console) (p. 622)
     • Control access to maintenance windows (AWS CLI) (p. 626)
     • Control access to maintenance windows (Tools for Windows PowerShell) (p. 630)

     To help you decide whether to use a custom service role or the Systems Manager service-linked role with a maintenance window task, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).

12. In the Input Parameters section, specify parameters for the document. For Automation documents, the system auto-populates some of the values. You can keep or replace these values.

   Important
   For Automation documents, you can optionally specify an Automation Assume Role. If you don't specify a role for this parameter, then the Automation workflow assumes the maintenance window service role you choose in step 11. As such, you must ensure that the maintenance window service role you choose has the appropriate AWS Identity and Access Management (IAM) permissions to perform the actions defined within the Automation document.
For example, the service-linked role for Systems Manager doesn't have the IAM permission `ec2:CreateSnapshot`, which is required to run the Automation document `AWS-CopySnapshot`. In this scenario, you must either use a custom maintenance window service role or specify an Automation Assume Role that has `ec2:CreateSnapshot` permissions. For information, see Getting started with Automation (p. 281).

13. Choose **Register Automation task**.

### Registering an Automation workflow task to a maintenance window (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to configure an Automation workflow as a registered task for a maintenance window.

**Before You Begin**

Before you complete the following procedure, you must create a maintenance window and register at least one target. For more information, see the following procedures:

- Step 1: Create the maintenance window (AWS CLI) (p. 642).
- Step 2: Register a target instance with the maintenance window (AWS CLI) (p. 643)

**To configure an Automation workflow as a registered task for a maintenance window**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already. For information, see Install or upgrade AWS command line tools (p. 57).
2. Create a command to configure an Automation workflow as a registered task for a maintenance window. Here are some template commands to help.

   **Linux**
   ```bash
   aws ssm register-task-with-maintenance-window \
   --window-id window_id \
   --name task_name \
   --task-arn document_name \
   --targets Key=targets,Values=value_1,value_2,value_3 \
   --service-role-arn service_role_arn \
   --task-type AUTOMATION \
   --task-invocation-parameters task_parameters_if_any \
   --priority task_priority \
   --max-concurrency a_number_of_instances_or_a_percentage_of_target_set \
   --max-errors a_number_of_errors_or_a_percentage_of_target_set
   ```

   **Note**
   If you configure an Automation workflow as a registered task by using the AWS CLI, use the `--Task-Invocation-Parameters` parameter to specify parameters to pass to a task when it runs. Don't use the `--Task-Parameters` parameter. The `--Task-Parameters` parameter is a legacy parameter.

   **Windows**
   ```bash
   aws ssm register-task-with-maintenance-window ^
   --window-id window_id ^
   --name task_name ^
   --task-arn document_name ^
   --targets Key=targets,Values=value_1,value_2,value_3 ^
   --service-role-arn service_role_arn ^
   --task-type AUTOMATION ^
   --task-invocation-parameters task_parameters_if_any ^
   --priority task_priority ^
   ```

329
--max-concurrency a_number_of_instances_or_a_percentage_of_target_set
--max-errors a_number_of_errors_or_a_percentage_of_target_set

**Note**
If you configure an Automation workflow as a registered task by using the AWS CLI, use the --Task-Invocation-Parameters parameter to specify parameters to pass to a task when it runs. Don't use the --Task-Parameters parameter. The --Task-Parameters parameter is a legacy parameter.

**PowerShell**

```powershell
Register-SSMTaskWithMaintenanceWindow `  
-WindowId window_id 
-Name "task_name" 
-TaskArn "document_name" 
-Target @{ Key="targets";Values="value_1" } 
-ServiceRoleArn "service_role_arn" 
-TaskType "AUTOMATION" 
-Automation_Parameter @{ "task_parameters_1"="task_parameter_1_value";"task_parameters_2"="task_parameter_2_value" } 
-Priority task_priority 
-MaxConcurrency a_number_of_instances_or_a_percentage_of_target_set 
-MaxError a_number_of_errors_or_a_percentage_of_target_set
```

**Note**
If you configure an Automation workflow as a registered task by using the AWS Tools for PowerShell, use the -Automation_Parameter parameter to specify parameters to pass to a task when the task runs. Don't use the -TaskParameters parameter. The -TaskParameters parameter is a legacy parameter.

The following example configures an Automation workflow as a registered task to a maintenance window with priority 1. The Automation workflow uses the AWS-StartEC2Instance document and the specified Automation assume role to start EC2 instances registered as targets to the maintenance window. The maintenance window runs the Automation workflow simultaneously on 5 instances maximum at any given time. Also, the registered task stops running on more instances for a particular execution interval if the error count exceeds 1.

**Linux**

```bash
aws ssm register-task-with-maintenance-window 
--window-id mw-0c50856d01EXAMPLE 
--name StartEC2Instances 
--task-arn AWS-StartEC2Instance 
--targets Key=WindowTargetIds,Values=e32eeeb2-646c-4f4b-8ed1-205fbEXAMPLE 
--service-role-arn arn:aws:iam::123456789012:role/MaintenanceWindowRole 
--task-type AUTOMATION 
--task-invocation-parameters "{"Automation":{"Parameters":{"InstanceId": ["{{TARGET_ID}}"],"AutomationAssumeRole":{"arn:aws:iam::123456789012:role/AutomationAssumeRole"}}}" 
--priority 1 
--max-concurrency 5 
--max-errors 1
```

**Windows**

```bash
aws ssm register-task-with-maintenance-window 
--window-id mw-0c50856d01EXAMPLE 
--name StartEC2Instances
```
Register-SSMTaskWithMaintenanceWindow `  
-WindowId mw-0c50858d01EXAMPLE  
-Name "StartEC2"  
-TaskArn "AWS-StartEC2Instance"  
-Target @{ Key="WindowTargetIds";Values="e32eeceb2-646c-4f4b-8ed1-205fbEXAMPLE" }  
-ServiceRoleArn "arn:aws:iam::123456789012:role/MaintenanceWindowRole"  
-TaskType "AUTOMATION"  
-Automation_Parameter @{ "InstanceId": "{TARGET_ID}";"AutomationAssumeRole":"arn:aws:iam::123456789012:role/AutomationAssumeRole" }  
-Priority 1  
-MaxConcurrency 5  
-MaxError 1

The command returns details for the new registered task similar to the following.

Linux

```
{
  "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

Windows

```
{
  "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

PowerShell

```
4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

3. To view the registered task, run the following command.

Linux

```
aws ssm describe-maintenance-window-tasks \
--window-id mw-0c50858d01EXAMPLE
```

Windows

```
aws ssm describe-maintenance-window-tasks \
--window-id mw-0c50858d01EXAMPLE
```
PowerShell

```
Get-SSMMaintenanceWindowTaskList -WindowId mw-0c50858d01EXAMPLE
```

The system returns information like the following.

**Linux**

```
{
    "Tasks": [
        {
            "ServiceRoleArn": "arn:aws:iam::123456789012:role/MaintenanceWindowRole",
            "MaxErrors": "1",
            "TaskArn": "AWS-StartEC2Instance",
            "MaxConcurrency": "5",
            "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
            "TaskParameters": {},
            "Priority": 0,
            "WindowId": "mw-0c50858d01EXAMPLE",
            "Type": "AUTOMATION",
            "Targets": [
                {
                    "Values": [
                        "e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
                    ],
                    "Key": "WindowTargetIds"
                }
            ],
            "Name": "StartEC2"
        }
    ]
}
```

**Windows**

```
{
    "Tasks": [
        {
            "ServiceRoleArn": "arn:aws:iam::123456789012:role/MaintenanceWindowRole",
            "MaxErrors": "1",
            "TaskArn": "AWS-StartEC2Instance",
            "MaxConcurrency": "5",
            "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
            "TaskParameters": {},
            "Priority": 0,
            "WindowId": "mw-0c50858d01EXAMPLE",
            "Type": "AUTOMATION",
            "Targets": [
                {
                    "Values": [
                        "e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
                    ],
                    "Key": "WindowTargetIds"
                }
            ],
            "Name": "StartEC2"
        }
    ]
}
```
Running Automation workflows by using different security models

This section includes information about how to run Automation workflows by using different security models.

Topics
- Running an Automation workflow as the current authenticated user (p. 333)
- Running an Automation workflow by using an IAM service role (p. 337)
- Running an Automation workflow by using delegated administration (p. 341)

Running an Automation workflow as the current authenticated user

The following procedures describe how to run an Automation workflow that runs in the context of the current AWS Identity and Access Management (IAM) user using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell. Running the Automation workflow in the context of the current IAM user means that you don’t need to configure additional IAM permissions as long as IAM user has permission to run the Automation document, or playbook, and any actions called by the document. If the IAM user has have administrator permissions in IAM, then you have permission to run this Automation.

Running an Automation workflow as the current authenticated user (console)

The following procedure describes how to use the Systems Manager console to run an Automation workflow as the current authenticated user.

To run the Automation document as the current authenticated user

2. In the navigation pane, choose Automation, and then choose Execute automation.
3. In the Automation document list, choose a document. Choose one or more options in the Document categories pane to filter SSM documents according to their purpose. To view a document that you own, choose the Owned by me tab. To view a document that is shared with your account, choose the Shared with me tab. To view all documents, choose the All documents tab.
   **Note**
   You can view information about a document by choosing the document name.
4. In the **Document details** section, verify that **Document version** is set to the version that you want to run. The system includes the following version options:

   - **Default version at runtime**: Choose this option if the Automation document is updated periodically and a new default version is assigned.
   - **Latest version at runtime**: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.
   - **1 (Default)**: Choose this option to run the first version of the document, which is the default.

5. Choose Next.

6. In the **Execution Mode** section, choose **Simple execution**.

   **Note**
   This procedure uses the **Simple execution** mode. However, you can alternatively choose **Rate control** or **Manual execution** and run the Automation workflow as the current authenticated user.

7. In the **Input parameters** section, specify the required inputs. To run the Automation workflow as the current authenticated user, do not specify an IAM service role for the value AutomationAssumeRole.

8. Choose **Execute**. The console displays the status of the Automation execution.

---

**Running an Automation workflow as the current authenticated user (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to run an Automation workflow as the current authenticated user.

**To run the Automation document as the current authenticated user**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see [Install or upgrade AWS command line tools](p. 57).

2. Run the following command to start an Automation workflow as the current authenticated user.

   **Linux**
   ```bash
   aws ssm start-automation-execution \
   --document-name DocumentName \
   --parameters ParametersRequiredByDocument
   ```

   **Windows**
   ```bash
   aws ssm start-automation-execution ^
   --document-name DocumentName ^
   --parameters ParametersRequiredByDocument
   ```

   **PowerShell**
   ```powershell
   Start-SSMAutomationExecution `^`  
   -DocumentName DocumentName `^`  
   -Parameter ParametersRequiredByDocument
   ```

   Here is an example using the document **AWS-RestartEC2Instance** to restart the specified EC2 instance.
Working with Automation executions

Linux

```bash
aws ssm start-automation-execution \
  --document-name "AWS-RestartEC2Instance" \
  --parameters "InstanceId=i-1234567890abcdef0"
```

Windows

```bash
aws ssm start-automation-execution ^
  --document-name "AWS-RestartEC2Instance" ^
  --parameters "InstanceId=i-1234567890abcdef0"
```

PowerShell

```bash
Start-SSMAutomationExecution `\`
  -DocumentName AWS-RestartEC2Instance `\`
  -Parameter @("InstanceId"="i-1234567890abcdef0")
```

The system returns information like the following.

Linux

```json
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

Windows

```json
{
  "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

PowerShell

```bash
4105a4fc-f944-11e6-9d32-0123456789ab
```

3. Run the following command to retrieve the status of the Automation workflow.

Linux

```bash
aws ssm describe-automation-executions \
  --filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

Windows

```bash
aws ssm describe-automation-executions ^
  --filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

PowerShell

```bash
Get-SSMAutomationExecutionList | `\`
  Where {$_AutomationExecutionId -eq "4105a4fc-f944-11e6-9d32-0123456789ab"}
```
The system returns information like the following.

**Linux**

```json
{
  "AutomationExecutionMetadataList": [
    {
      "AutomationExecutionStatus": "InProgress",
      "CurrentStepName": "stopInstances",
      "Outputs": {},
      "DocumentName": "AWS-RestartEC2Instance",
      "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab",
      "DocumentVersion": "1",
      "ResolvedTargets": {
        "ParameterValues": [],
        "Truncated": false
      },
      "AutomationType": "Local",
      "Mode": "Auto",
      "ExecutionStartTime": 1564600648.159,
      "CurrentAction": "aws:changeInstanceState",
      "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
      "LogFile": "",
      "Targets": []
    }
  ]
}
```

**Windows**

```json
{
  "AutomationExecutionMetadataList": [
    {
      "AutomationExecutionStatus": "InProgress",
      "CurrentStepName": "stopInstances",
      "Outputs": {},
      "DocumentName": "AWS-RestartEC2Instance",
      "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab",
      "DocumentVersion": "1",
      "ResolvedTargets": {
        "ParameterValues": [],
        "Truncated": false
      },
      "AutomationType": "Local",
      "Mode": "Auto",
      "ExecutionStartTime": 1564600648.159,
      "CurrentAction": "aws:changeInstanceState",
      "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
      "LogFile": "",
      "Targets": []
    }
  ]
}
```

**PowerShell**

```plaintext
AutomationExecutionId       : 4105a4fc-f944-11e6-9d32-0123456789ab
AutomationExecutionStatus  : InProgress
```
Running an Automation workflow by using an IAM service role

The following procedures describe how to use the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell to run an Automation workflow using an AWS Identity and Access Management (IAM) service role (or *assume role*). The service role gives the Automation workflow permission to perform actions on your behalf. Configuring a service role is useful when you want to restrict permissions and run actions with least privilege. This is useful, for example, when you want to restrict a user's privileges on a resource, such as an EC2 instance, but you want to allow the user to run an Automation workflow that performs a specific set of actions. In this scenario, you can create a service role with elevated privileges and allow the user to run the Automation workflow.

**Before You Begin**

Before you complete the following procedures, you must create the IAM service role and configure a trust relationship for Automation. For more information, see *Task 1: Create a service role for Automation* (p. 285).

**Running an Automation workflow by using an IAM service role (console)**

The following procedure describes how to use the Systems Manager console to run an Automation workflow that uses an IAM service role (or *assume role*).

**To run an Automation workflow using a service role**

2. In the navigation pane, choose Automation, and then choose Execute automation.
3. In the Automation document list, choose a document. Choose one or more options in the Document categories pane to filter SSM documents according to their purpose. To view a document that you own, choose the Owned by me tab. To view a document that is shared with your account, choose the Shared with me tab. To view all documents, choose the All documents tab.

   **Note**
   You can view information about a document by choosing the document name.

4. In the Document details section, verify that Document version is set to the version that you want to run. The system includes the following version options:

   - **Default version at runtime**: Choose this option if the Automation document is updated periodically and a new default version is assigned.
• **Latest version at runtime**: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.

• **1 (Default)**: Choose this option to run the first version of the document, which is the default.

5. Choose **Next**.

6. In the **Execution Mode** section, choose **Simple execution**.

   **Note**
   
   This procedure uses the **Simple execution** mode. However, you can alternatively choose **Rate control**, **Multi-account and Region**, or **Manual execution** and run the Automation workflow using a service role.

7. In the **Input parameters** section, specify the required inputs. In the **Automation Assume Role** box, paste the ARN of the IAM service role.

8. Choose **Execute**. The console displays the status of the Automation execution.

---

**Running an Automation workflow by using an IAM service role (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to run an Automation workflow that uses an IAM service role (or **assume role**).

**To run an Automation workflow using a service role**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to start an Automation workflow that uses an IAM service role.

   **Linux**

   ```bash
   aws ssm start-automation-execution \
   --document-name DocumentName \
   --parameters "ParametersRequiredByDocument","AutomationAssumeRole=arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"
   ```

   **Windows**

   ```bash
   aws ssm start-automation-execution ^
   --document-name DocumentName ^
   --parameters "ParametersRequiredByDocument","AutomationAssumeRole=arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"
   ```

   **PowerShell**

   ```powershell
   Start-SSMAutomationExecution `\`
   -DocumentName DocumentName `\`
   -Parameter @{
     "ParametersRequiredByDocument"="ParameterValues";
     "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"}
   ```

   Here is an example using the document **AWS-RestartEC2Instance** to restart the specified EC2 instance using the IAM service role **AmazonSSMAutomationRole**.
AWS Systems Manager User Guide
Working with Automation executions

Linux

```bash
aws ssm start-automation-execution \
   --document-name "AWS-RestartEC2Instance" \
   --parameters
"InstanceId=i-1234567890abcdef0","AutomationAssumeRole=arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"
```

Windows

```bash
aws ssm start-automation-execution ^
   --document-name "AWS-RestartEC2Instance" ^
   --parameters
"InstanceId=i-1234567890abcdef0","AutomationAssumeRole=arn:aws:iam::123456789012:role/AmazonSSMAutomationRole"
```

PowerShell

```powershell
Start-SSMAutomationExecution `\n   -DocumentName "AWS-RestartEC2Instance" `\n   -Parameter @{$\n   
   "InstanceId"="i-1234567890abcdef0"; `\n   "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole"} `\n```

The system returns information like the following.

Linux

```json
{
   "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

Windows

```json
{
   "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab"
}
```

PowerShell

```
4105a4fc-f944-11e6-9d32-0123456789ab
```

3. Run the following command to retrieve the status of the Automation workflow.

Linux

```bash
aws ssm describe-automation-executions \
   --filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```

Windows

```bash
aws ssm describe-automation-executions ^
   --filter "Key=ExecutionId,Values=4105a4fc-f944-11e6-9d32-0123456789ab"
```
PowerShell

```powershell
Get-SSMAutomationExecutionList | ` Where {$_AutomationExecutionId -eq "4105a4fc-f944-11e6-9d32-0123456789ab"}
```

The system returns information like the following.

**Linux**

```json
{
  "AutomationExecutionMetadataList": [
    {
      "AutomationExecutionStatus": "InProgress",
      "CurrentStepName": "stopInstances",
      "Outputs": {},
      "DocumentName": "AWS-RestartEC2Instance",
      "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab",
      "DocumentVersion": "1",
      "ResolvedTargets": {
        "ParameterValues": [],
        "Truncated": false
      },
      "AutomationType": "Local",
      "Mode": "Auto",
      "ExecutionStartTime": 1564600648.159,
      "CurrentAction": "aws:changeInstanceState",
      "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
      "LogFile": "",
      "Targets": []
    }
  ]
}
```

**Windows**

```json
{
  "AutomationExecutionMetadataList": [
    {
      "AutomationExecutionStatus": "InProgress",
      "CurrentStepName": "stopInstances",
      "Outputs": {},
      "DocumentName": "AWS-RestartEC2Instance",
      "AutomationExecutionId": "4105a4fc-f944-11e6-9d32-0123456789ab",
      "DocumentVersion": "1",
      "ResolvedTargets": {
        "ParameterValues": [],
        "Truncated": false
      },
      "AutomationType": "Local",
      "Mode": "Auto",
      "ExecutionStartTime": 1564600648.159,
      "CurrentAction": "aws:changeInstanceState",
      "ExecutedBy": "arn:aws:sts::123456789012:assumed-role/Administrator/Admin",
      "LogFile": "",
      "Targets": []
    }
  ]
}
```
PowerShell

For more examples of how to use Systems Manager Automation, see Automation walkthroughs (p. 568). For information about how to get started with Automation, see Getting started with Automation (p. 281).

Running an Automation workflow by using delegated administration

When you run an AWS Systems Manager Automation workflow, by default, the Automation runs in the context of the AWS Identity and Access Management (IAM) user who initiated the execution. This means, for example, if your IAM user account has administrator permissions, then the Automation runs with administrator permissions and full access to the resources being configured by the Automation workflow. As a security best practice, we recommend that you run Automation workflows by using an IAM service role (also called an assumed role) that is configured with the AmazonSSMAutomationRole managed policy. Using an IAM service role to run Automation is called delegated administration.

When you use a service role, the Automation workflow is allowed to run against the AWS resources, but the user who ran the Automation has restricted access (or no access) to those resources. For example, you can configure a service role and use it with Automation to restart one or more EC2 instances. The Automation workflow restarts the instances, but the service role does not give the user permission to access those instances.

You can specify a service role at runtime when you run an Automation workflow, or you can create custom Automation documents and specify the service role directly in the document. If you specify a service role, either at runtime or in an Automation document, then the service runs in the context of the specified service role. If you don't specify a service role, then the system creates a temporary session in the context of the user and runs the Automation.

Note
You must specify a service role for Automation workflows that you expect to run longer than 12 hours. If you start a long-running Automation in the context of a user, the user's temporary session expires after 12 hours.

Delegated administration ensures elevated security and control of your AWS resources. It also enables an enhanced auditing experience because actions are being performed against your resources by a central service role instead of multiple IAM accounts.
To properly illustrate how delegated administration can work in an organization, this topic describes the following tasks as though these tasks were performed by three different people in an organization:

- Create a test IAM user account called AutomationRestrictedOperator (Administrator).
- Create an IAM service role for Automation (Administrator).
- Create a simple Automation document (based on a preexisting Automation document) that specifies the service role (SSM Document Author).
- Run the Automation as the test user (Restricted Operator).

In some organizations, all three of these tasks are performed by the same person, but identifying the different roles here shows how delegated administration enables enhanced security in complex organizations.

**Important**
As a security best practice, we recommend that you always use a service role to run Automation workflows, even if you are an administrator who performs all of these tasks.

The procedures in this section link to topics in other AWS guides or other Systems Manager topics. We recommend that you open links to other topics in a new tab in your web browser so you don't lose your place in this topic.

**Topics**
- Create a test user account (p. 342)
- Create an IAM service role for Automation (p. 343)
- Create a custom Automation document (p. 343)
- Run the custom Automation document (p. 344)

**Create a test user account**

This section describes how to create an IAM test user account with restricted permissions. The permissions set allows the user to run Automation workflows, but the user doesn't have access to the AWS resources targeted by Automation. The operator can also view the results of the Automation workflows. You start by creating the custom IAM permissions policy, and then you create the user account and assign permissions to it.

**Create an IAM test user**

1. Create a permissions policy named OperatorRestrictedPermissions. For information about how to create a new IAM permissions policy, see Create an IAM Policy (Console) in the IAM User Guide. Create the policy on the JSON tab, and specify the following permissions set.

```json
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect":"Allow",
         "Action": [
            "ssm:DescribeAutomationExecutions",
            "ssm:DescribeAutomationStepExecutions",
            "ssm:DescribeDocument",
            "ssm:GetAutomationExecution",
            "ssm:GetDocument",
            "ssm:GetDocument",
            "ssm:GetDocument",
            "ssm:GetDocument",
            "ssm:GetDocumentVersions",
            "ssm:StartAutomationExecution"
         ],
         "Resource": "*"
      }
   ]
}
```
2. Create a new IAM user account named AutomationRestrictedOperator. For information about how to create a new IAM user, see Creating IAM Users (Console) in the IAM User Guide. When prompted, choose **Attach existing policies directly**, and choose the policy you just created.

3. Note the user name, password, and the **Console login link**. You will log into this account later in this topic.

Create an IAM service role for Automation

The following procedure links to other topics to help you create the service role and to configure Automation to trust this role.

To create the service role and enable Automation to trust it

1. Create the Automation service role. For information, see Task 1: Create a service role for Automation (p. 283).
2. Note the service role Amazon Resource Name (ARN). You will specify this ARN in the next procedure.

Create a custom Automation document

This section describes how to create a custom Automation document that restarts EC2 instances. AWS provides a default SSM document for restarting instances called AWS-RestartEC2Instance. The following procedure copies the content of that document to show you how to enter the service role in a document when you create your own. By specifying the service role directly in the document, the user running the document does not require iam:PassRole permissions. Without iam:PassRole permissions, the user can't use the service role elsewhere in AWS.

To create a custom Automation document

2. In the navigation pane, choose **Documents**.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Documents** in the navigation pane.
3. Choose **Create document**.
4. In the **Name** field, type a name for the document, such as Restart-EC2InstanceDemo.
5. In the **Document type** list, choose **Automation document**.
6. In the **Content** section, choose **JSON**, and then paste the following content. Replace **AssumeRoleARN** with the ARN of the service role you created in the previous procedure.

```json
{
   "description": "Restart EC2 instances(s)",
   "schemaVersion": "0.3",
   "assumeRole": "AssumeRoleARN",
   "parameters": {
      "InstanceId": {
         "type": "StringList",
         "description": "(Required) EC2 Instance to restart"
      }
   },
   "mainSteps": [
      {
```
Run the custom Automation document

The following procedure describes how to run the document you just created using the restricted operator role you created earlier in this topic. The user can run the document you created earlier because their IAM account permissions enable them to see and run the document. The user can’t, however, log on to the instances that you will restart with this Automation workflow.

1. In the https://console.aws.amazon.com/ec2/, copy the instance IDs for one or more instances that you want to restart by using the following Automation workflow.
2. Sign out of the AWS Management Console, and then sign back in by using the test user account Console login link that you copied earlier.
4. In the navigation pane, choose Automation.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Automation.
5. Choose Execute automation.
6. Choose the custom Automation document you created earlier in this topic.
7. In the Document details section, verify that Document version is set to 1 (Default).
8. Choose Next.
9. In the Execution mode section, choose Simple execution.
10. In the Input parameters section, type one or more instance IDs that you want to restart, and then choose Execute.

Execution details describes the status of the Automation. Step 1 stops the instances. Step 2 starts the instances.

Running Automation workflows in multiple AWS Regions and accounts

You can run AWS Systems Manager Automations across multiple AWS Regions and AWS accounts or AWS Organizational Units (OUs) from an Automation management account. Running Automations in multiple Regions and accounts or OUs reduces the time required to administer your AWS resources while enhancing the security of your computing environment.
For example, you can centrally implement patching and security updates, remediate compliance drift on VPC configurations or S3 bucket policies, and manage resources, such as EC2 instances, at scale. The following graphic shows an example of a user who is running the AWS-RestartEC2Instances document in multiple Regions and accounts from an Automation management account. The Automation locates the instances by using the specified tags in the specified Regions and accounts.

**Note**

When you run an Automation across multiple Regions and accounts, you target resources by using tags or the name of an AWS resource group. The resource group must exist in each target account and Region, and the resource group name must be the same in each target account and Region. The Automation fails to run on those resources that don’t have the specified tag or that aren’t included in the specified resource group.

**Important**

Your account is charged for running Automations in multiple Regions and accounts. Multi-Region and account step executions are considered special steps. There is no step limit for special steps, but your account is charged for each step processed by Systems Manager. For more information, see the [AWS Systems Manager Pricing](https://aws.amazon.com/pricing/) page.

**How It Works**

Running Automations across multiple Regions and accounts or OUs works as follows:

1. Verify that all resources on which you want to run the Automation, in all Regions and accounts or OUs, use identical tags. If they don’t, you can add them to an AWS resource group and target that group. For more information, see [What Is AWS Resource Groups?](https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-automation-resource-groups.html)

2. Sign in to the AWS Identity and Access Management (IAM) account that you want to configure as the Automation Master account.

3. Use the procedure in this topic to create the IAM execution role named `AWS-SystemsManager-AutomationExecutionRole`. This role gives the user permission to run Automation workflows.

4. Use the procedure in this topic to create the second IAM role, named `AWS-SystemsManager-AutomationAdministrationRole`. This role gives the user permission to run Automation workflows in multiple AWS accounts and OUs.
5. Choose the Automation document (playbook), Regions, and accounts or OUs where you want to run the Automation workflow.

**Note**
Automations do not execute recursively through OUs. Be sure the target OU contains the desired accounts.

6. Run the Automation.

7. Use the `GetAutomationExecution`, `DescribeAutomationStepExecutions`, and `DescribeAutomationExecutions` API actions from the AWS Systems Manager console or the AWS CLI to monitor workflow progress.

**Setting up management account permissions for multi-Region and multi-account Automation execution**

Use the following procedure to create the required IAM roles for Systems Manager Automation multi-Region and multi-account execution by using AWS CloudFormation. This procedure describes how to create the `AWS-SystemsManager-AutomationExecutionRole` role. You must create this role in every account that you want to target to run multi-Region and multi-account Automations.

This procedure also describes how to create the `AWS-SystemsManager-AutomationAdministrationRole` role. You only need to create this role in the Automation management account.

**To create the required IAM roles for multi-Region and multi-account Automation executions by using AWS CloudFormation**

1. Download and unzip the `AWS-SystemsManager-AutomationExecutionRole.zip` folder. This folder includes the `AWS-SystemsManager-AutomationExecutionRole.json` AWS CloudFormation template file.

   **Note**
   We recommend not changing the role name as specified in the template to something besides `AWS-SystemsManager-AutomationExecutionRole`. Otherwise, your multi-Region and multi-Account Automation workflows might fail.


3. Choose Create Stack.

4. In the Choose a template section, choose Upload a template to Amazon S3.

5. Choose Browse, and then choose the `AWS-SystemsManager-AutomationExecutionRole.json` AWS CloudFormation template file.

6. Choose Next.

7. On the Specify Details page, in the Stack Name field, enter a name.

8. In the Parameters section, in the MasterAccountID field, enter the ID for the account that you want to use to run multi-Region and multi-account Automations.

9. Choose Next.

10. On the Options page, enter values for any options you want to use. Choose Next.

11. On the Review page, scroll down and choose the I acknowledge that AWS CloudFormation might create IAM resources option.

12. Choose Create.

   AWS CloudFormation shows the CREATE_IN_PROGRESS status for approximately three minutes. The status changes to CREATE_COMPLETE.

13. Repeat this procedure in every account that you want to target to run multi-Region and multi-account Automations.

14. Download the `AWS-SystemManager-AutomationAdministrationRole.zip` folder and repeat this procedure for the `AWS-SystemManager-AutomationAdministrationRole` role. You only need
to create the **AWS-SystemManager-AutomationAdministrationRole** role in the Automation management account.

**Note**
The IAM user or role you use to run a multi-Region or multi-account Automation must have the `iam:PassRole` permission for the **AWS-SystemManager-AutomationAdministrationRole** role. We recommend not changing the role name as specified in the template to something besides **AWS-SystemManager-AutomationAdministrationRole**. Otherwise, your multi-Region and multi-account Automation workflows might fail.

### Run an Automation in multiple Regions and accounts (console)

The following procedure describes how to use the Systems Manager console to run an Automation in multiple Regions and accounts from the Automation management account.

**Before You Begin**

Before you complete the following procedure, note the following information:

- **AWS account IDs or OUs** where you want to run the Automation.
- **Regions supported by Systems Manager** where you want to run the Automation.
- **The tag key and the tag value**, or the name of the resource group, where you want to run the Automation.

**To run an Automation workflow in multiple Regions and accounts**

2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the **Automation document** list, choose a document. Choose one or more options in the **Document categories** pane to filter SSM documents according to their purpose. To view a document that you own, choose the **Owned by me** tab. To view a document that is shared with your account, choose the **Shared with me** tab. To view all documents, choose the **All documents** tab.

    **Note**
    You can view information about a document by choosing the document name.

4. In the **Document details** section, verify that **Document version** is set to the version that you want to run. The system includes the following version options:

    - **Default version at runtime**: Choose this option if the Automation document is updated periodically and a new default version is assigned.
    - **Latest version at runtime**: Choose this option if the Automation document is updated periodically, and you want to run the version that was most recently updated.
    - **1 (Default)**: Choose this option to run the first version of the document, which is the default.

5. Choose **Next**.
6. On the **Execute automation document** page, choose **Multi-account and Region**.
7. In the **Target accounts and Regions** section, use the **Accounts and organizational (OUs)** field to specify the different AWS accounts or AWS Organizational Units (OUs) where you want to run the Automation. Separate multiple accounts or OUs with a comma.
8. Use the **AWS Regions** list to choose one or more Regions where you want to run the Automation.
9. Use the **Multi-Region and account rate control** options to restrict the Automation execution to a limited number of accounts running in a limited number of Regions. These options don't restrict the number of AWS resources that can run the Automations.
a. In the **Location (account-Region pair) concurrency** section, choose an option to restrict the number of Automation workflows that can run in multiple accounts and Regions at the same time. For example, if you choose to run an Automation in five (5) AWS accounts, which are located in four (4) AWS Regions, then Systems Manager runs Automations in a total of 20 account-Region pairs. You can use this option to specify an absolute number, such as 2, so that the Automation only runs in two account-Region pairs at the same time. Or you can specify a percentage of the account-Region pairs that can run at the same time. For example, with 20 account-Region pairs, if you specify 20%, then the Automation simultaneously runs in a maximum of five (5) account-Region pairs.

- Choose **targets** to enter an absolute number of account-Region pairs that can run the Automation workflow simultaneously.
- Choose **percent** to enter a percentage of the total number of account-Region pairs that can run the Automation workflow simultaneously.

b. In the **Error threshold** section, choose an option:

- Choose **errors** to enter an absolute number of errors allowed before Automation stops sending the workflow to other resources.
- Choose **percent** to enter a percentage of errors allowed before Automation stops sending the workflow to other resources.

10. In the **Targets** section, choose how you want to target the AWS Resources where you want to run the Automation. These options are required.

a. Use the **Parameter** list to choose a parameter. The items in the **Parameter** list are determined by the parameters in the Automation document that you selected at the start of this procedure. By choosing a parameter you define the type of resource on which the Automation workflow runs.

b. Use the **Targets** list to choose how you want to target resources.

   i. If you chose to target resources by using parameter values, then enter the parameter value for the parameter you chose in the **Input parameters** section.

   ii. If you chose to target resources by using AWS Resource Groups, then choose the name of the group from the **Resource Group** list.

   iii. If you chose to target resources by using tags, then enter the tag key and (optionally) the tag value in the fields provided. Choose **Add**.

   iv. If you want to run an Automation playbook on all instances in the current AWS account and Region, then choose **All instances**.

11. In the **Input parameters** section, specify the required inputs. Optionally, you can choose an IAM service role from the **AutomationAssumeRole** list.

    **Note**
    You may not need to choose some of the options in the **Input parameters** section. This is because you targeted resources in multiple Regions and accounts by using tags or a resource group. For example, if you chose the AWS-RestartEC2Instance document, then you don't need to specify or choose instance IDs in the **Input parameters** section. The Automation execution locates the instances to restart by using the tags you specified.

12. Use the options in the **Rate control** section to restrict the number of AWS resources that can run the Automation within each account-Region pair.

   In the **Concurrency** section, choose an option:

   - Choose **targets** to enter an absolute number of targets that can run the Automation workflow simultaneously.
   - Choose **percentage** to enter a percentage of the target set that can run the Automation workflow simultaneously.
13. In the **Error threshold** section, choose an option:
   - Choose **errors** to enter an absolute number of errors allowed before Automation stops sending the workflow to other resources.
   - Choose **percentage** to enter a percentage of errors allowed before Automation stops sending the workflow to other resources.

14. Choose **Execute**.

### Run an Automation in multiple Regions and accounts (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to run an Automation in multiple Regions and accounts from the Automation management account.

**Before You Begin**

Before you complete the following procedure, note the following information:

- AWS account IDs or OUs where you want to run the Automation.
- Regions supported by Systems Manager where you want to run the Automation.
- The tag key and the tag value, or the name of the resource group, where you want to run the Automation.

**To run an Automation workflow in multiple Regions and accounts**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Use the following format to create a command to run an Automation workflow in multiple Regions and accounts.

   **Linux**
   ```shell
   aws ssm start-automation-execution
   --document-name name_of_Automation_document
   --parameters
   AutomationAssumeRole=arn:aws:iam::Automation_management_account_ID:role/AWS-SystemsManager-AutomationAdministrationRole
   --target-parameter-name parameter_name (required)
   --targets Key=tag_key,Values=tag_value
   --target-locations
   Accounts=account_ID_1,account_ID_2,account_ID_3,Regions=Region_1,Region_2,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
   ```

   **Windows**
   ```shell
   aws ssm start-automation-execution
   --document-name name_of_Automation_document
   --parameters
   AutomationAssumeRole=arn:aws:iam::Automation_management_account_ID:role/AWS-SystemsManager-AutomationAdministrationRole
   --target-parameter-name parameter_name (required)
   --targets Key=tag_key,Values=tag_value
   --target-locations
   Accounts=account_ID_1,account_ID_2,account_ID_3,Regions=Region_1,Region_2,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
   ```
PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "target_key"
$Targets.Values = "target_value"

Start-SSMAutomationExecution
-DocumentName "name_of_Automation_document"
-Parameter @(
  "AutomationAssumeRole"="arn:aws:iam::Automation_management_account_ID:role/AWS-SystemsManager-AutomationAdministrationRole"
) `-TargetParameterName "parameter_name (required)"
-Target $Targets
-TargetLocation @{
  "Accounts"="account_ID_1","account_ID_2","account_ID_3";
  "Regions"="Region_1","Region_2";
  "ExecutionRoleName"="AWS-SystemsManager-AutomationExecutionRole"
}
```

Here are a few examples.

**Example 1:** This example restarts EC2 instances in the 123456789012 and 987654321098 accounts, which are located in the us-east-2 and us-west-1 Regions. The instances must be tagged with the tag key-pair value Env-PROD.

**Linux**

```bash
aws ssm start-automation-execution
--document-name AWS-RestartEC2Instance
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole
--target-parameter-name InstanceId
--targets Key=tag:Env,Values=PROD
--target-locations Accounts=123456789012,987654321098,Regions=us-east-2,us-west-1,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

**Windows**

```bash
aws ssm start-automation-execution
--document-name AWS-RestartEC2Instance
--parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole
--target-parameter-name InstanceId
--targets Key=tag:Env,Values=PROD
--target-locations Accounts=123456789012,987654321098,Regions=us-east-2,us-west-1,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

**PowerShell**

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "tag:Env"
$Targets.Values = "PROD"

Start-SSMAutomationExecution
-DocumentName "AWS-RestartEC2Instance"
-Parameter @(
  "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole"
) `-TargetParameterName "InstanceId"
-Target $Targets
```
Example 2: This example restarts EC2 instances in the 123456789012 and 987654321098 accounts, which are located in the eu-central-1 Region. The instances must be members of the prod-instances AWS resource group.

Linux

```bash
aws ssm start-automation-execution \
   --document-name AWS-RestartEC2Instance \
   --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole \
   --target-parameter-name InstanceId \
   --targets Key=ResourceGroup,Values=prod-instances \
   --target-locations Accounts=123456789012,987654321098,Regions=eu-central-1,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

Windows

```bash
aws ssm start-automation-execution ^
   --document-name AWS-RestartEC2Instance ^
   --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole ^
   --target-parameter-name InstanceId ^
   --targets Key=ResourceGroup,Values=prod-instances ^
   --target-locations Accounts=123456789012,987654321098,Regions=eu-central-1,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "prod-instances"

Start-SSMAutomationExecution ` 
   -DocumentName "AWS-RestartEC2Instance" ` 
   -Parameter @{ 
   "AutomationAssumeRole"="arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole" } ` 
   -TargetParameterName "InstanceId" ` 
   -Target $Targets ` 
   -TargetLocation @{$ 
   "Accounts"="123456789012","987654321098"; 
   "Regions"="eu-central-1"; 
   "ExecutionRoleName"="AWS-SystemsManager-AutomationExecutionRole" } ` 
```

Example 3: This example restarts EC2 instances in the ou-1a2b3c-4d5e6c AWS organizational unit (OU). The instances are located in the us-west-1 and us-west-2 Regions. The instances must be members of the WebServices AWS resource group.

Linux

```bash
aws ssm start-automation-execution \
   --document-name AWS-RestartEC2Instance \
```
Working with Automation executions

Windows

```bash
aws ssm start-automation-execution ^
  --document-name AWS-RestartEC2Instance ^
  --parameters AutomationAssumeRole=arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole ^
  --target-parameter-name InstanceId ^
  --targets Key=ResourceGroup,Values=WebServices ^
  --target-locations Accounts=ou-1a2b3c-4d5e6c,Regions=us-west-1,us-west-2,ExecutionRoleName=AWS-SystemsManager-AutomationExecutionRole
```

PowerShell

```powershell
$Targets = New-Object Amazon.SimpleSystemsManagement.Model.Target
$Targets.Key = "ResourceGroup"
$Targets.Values = "WebServices"
Start-SSMAutomationExecution -DocumentName "AWS-RestartEC2Instance" -Parameter @{
  "AutomationAssumeRole" = "arn:aws:iam::123456789012:role/AWS-SystemsManager-AutomationAdministrationRole"
} -TargetParameterName "InstanceId" -Target $Targets -TargetLocation @{
  "Accounts" = "ou-1a2b3c-4d5e6c";
  "Regions" = "us-west-1";
  "ExecutionRoleName" = "AWS-SystemsManager-AutomationExecutionRole"
}
```

The system returns information similar to the following.

Linux

```json
{
  "AutomationExecutionId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

Windows

```json
{
  "AutomationExecutionId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
```

PowerShell

```powershell
4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

3. Run the following command to view the workflow execution.
4. Run the following command to view details about the execution progress.

**Linux**

```
aws ssm get-automation-execution \
   --automation-execution-id 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

**Windows**

```
aws ssm get-automation-execution ^
   --automation-execution-id 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
```

**PowerShell**

```
Get-SSMAutomationExecution `^
   -AutomationExecutionId a4a3c0e9-7efd-462a-8594-01234EXAMPLE
```

**Note**

You can also monitor the status of the workflow in the console. In the execution list, choose the execution you just ran and then choose the **Steps** tab. This tab shows the status of the workflow actions.

**Related content**

**Centralized multi-account and multi-Region patching with AWS Systems Manager Automation**

**Working with Automation documents (Playbooks)**

A Systems Manager Automation document, or *operational playbook*, defines the *actions* that Systems Manager performs on your managed instances and other AWS resources when an automation execution runs. A document contains one or more steps that run in sequential order. Each step is built around a single action. Output from one step can be used as input in a later step.

The process of running these actions and their steps is called the *automation workflow*.

**Action types** supported for Automation documents let you automate a wide variety of operations in your AWS environment. For example, using the `executeScript` action type, you can embed a python or PowerShell script directly in your workflow. (When you create a custom Automation document, you
can add your script inline, or attach it from an S3 bucket or from your local machine.) You can automate management of your AWS CloudFormation resources by using the `createStack` and `deleteStack` action types. In addition, using the `executeAwsApi` action type, a step can run any API operation in any AWS service, including creating or deleting AWS resources, starting other processes, triggering notifications, and many more.

For a list of all 20 supported action types for Automation steps, see `Systems Manager Automation actions reference (p. 386)`.

AWS Systems Manager Automation provides several documents with pre-defined steps that you can use to perform common tasks like restarting one or more EC2 instances or creating an Amazon Machine Image (AMI). You can also create your own Automation documents and share them with other AWS accounts, or make them public for all Automation users.

Automation documents are written using JavaScript Object Notation (JSON) or YAML. Using the Document Builder in the Systems Manager Automation console, however, you can create an Automation document without having to author in native JSON or YAML.

**Important**

If you run an automation workflow that invokes other services by using an AWS Identity and Access Management (IAM) service role, be aware that the service role must be configured with permission to invoke those services. This requirement applies to all AWS Automation documents (AWS-* documents) such as the `AWS-ConfigureS3BucketLogging`, `AWS-CreateDynamoDBBackup`, and `AWS-RestartEC2Instance` documents, to name a few. This requirement also applies to any custom Automation documents you create that invoke other AWS services by using actions that call other services. For example, if you use the `aws:executeAwsApi`, `aws:createStack`, or `aws:copyImage` actions, then you must configure the service role with permission to invoke those services. You can enable permissions to other AWS services by adding an IAM inline policy to the role. For more information, see *(Optional) add an Automation inline policy to invoke other AWS services (p. 284).*

For information about the actions that you can specify in a Systems Manager Automation document, see `Systems Manager Automation actions reference (p. 386)`.

For information about the AWS managed Automation documents that run scripts, see `Amazon managed Automation documents that run scripts (p. 364)`.

For information about using Document Builder to create a custom Automation document, see `Creating Automation documents using Document Builder (p. 355)`.

For information about creating custom Automation documents that run scripts, see the following topics:

- **Creating Automation documents that run scripts (p. 358)** – Provides information for using Document Builder to create an Automation document that includes the `aws:executeScript` action.
- **Creating an Automation document that runs scripts (command line) (p. 361)** – Provides information for using a command line tool to create an Automation document that runs a script.
- **Walkthrough: Using Document Builder to create a custom Automation document (p. 604)** – Provides step-by-step guidance for creating an Automation document that runs scripts to (1) launch an EC2 instance and (2) wait for the instance status to change to ok.

**Contents**

- Creating Automation documents using Document Builder (p. 355)
- Creating an Automation document using the Editor (p. 357)
- Creating Automation documents that run scripts (p. 358)
- Creating dynamic Automation workflows with conditional branching (p. 365)
Creating Automation documents using Document Builder

If the AWS Systems Manager public Automation documents, or playbooks, don't support all the actions you want to perform on your AWS resources, you can create your own documents. To create a custom Automation document, you can manually create a local JavaScript Object Notation (JSON) or YAML format file with the appropriate automation actions. Alternatively, you can use the Document Builder in the Systems Manager console to more easily build a custom playbook-style document.

Using the Document Builder, you can add Automation action steps to your custom document and provide the required parameters without having to use JSON or YAML syntax. After you add steps and create the document, the system converts the actions you've added into the YAML format that Systems Manager can use to run automation workflows.

Automation documents support the use of Markdown, a markup language, which allows you to add wiki-style descriptions to documents and individual steps within the document. For more information on using Markdown, see Using Markdown in AWS.

**Tip**
This topic provides general information for using Document Builder with any supported action type. For more information about creating Automation documents that run scripts, see the following topics:

- **Creating Automation documents that run scripts (p. 358)** – Provides information for using Document Builder to create an Automation document that includes the `aws:executeScript` action.
- **Creating an Automation document that runs scripts (command line) (p. 361)** – Provides information for using a command line tool to create an Automation document that runs a script.
- **Walkthrough: Using Document Builder to create a custom Automation document (p. 604)** – Provides step-by-step guidance for creating an Automation document that runs scripts to (1) launch an Amazon Elastic Compute Cloud (EC2) instance and (2) wait for the instance status to change to `ok`.

**Before You Begin**

Before you create a custom Automation document using Document Builder, we recommend that you read about the different actions that you can use within an Automation document. For more information see, Systems Manager Automation actions reference (p. 386).

**To create an Automation document using Document Builder**

2. In the navigation pane, choose **Documents**.

   - **or**

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Documents** in the navigation pane.
3. Choose **Create automation**.
4. For **Name**, type a descriptive name for the document.
5. For **Document description**, provide the markdown style description for the document. You can provide instructions for using the document, numbered steps, or any other type of information to describe the document. Refer to the default text for information about formatting your content.

   **Tip**
   Toggle between **Hide preview** and **Show preview** to see what your description content looks like as you compose.

6. (Optional) For **Assume role**, enter the name or ARN of a service role to perform actions on your behalf. If you don't specify a role, Automation uses the access permissions of the user who invokes the automation execution.

   **Important**
   For Automation documents not owned by Amazon that use the `aws:executeScript` action, a role must be specified. For information, see Permissions for running Automation executions (p. 358).

7. (Optional) For **Outputs**, enter any outputs for the execution of this document to make available for other processes.

   For example, if your document creates a new AMI, you might specify "["CreateImage.ImageId"]", and then use this output to create new instances in a subsequent automation execution.

8. (Optional) Expand the **Input parameters** section and do the following.

   1. For **Parameter name**, enter a descriptive name for the document parameter you are creating.
   2. For **Type**, choose a type for the parameter, such as `String` or `MapList`.
   3. For **Required**, do one of the following:
      - Choose **Yes** if a value for this document parameter must be supplied at runtime.
      - Choose **No** if the parameter is not required, and (optional) enter a default parameter value in **Default value**.
   4. For **Description**, enter a description for the document parameter.

   **Note**
   To add more document parameters, choose **Add a parameter**. To remove a document parameter, choose the **X** (Remove) button.

9. (Optional) Expand the **Target type** section and choose a target type to define the kinds of resources the document can run on. For example, to run a document on EC2 instances, choose `/ AWS::EC2::Instance`.

   **Note**
   If you specify a value of `/`, the document can run on all types of resources. For a list of valid resource types, see AWS Resource Types Reference in the AWS CloudFormation User Guide.

10. (Optional) Expand the **Document tags** section and enter one or more tag key-value pairs to apply to the document. Tags make it easier to identify, organize, and search for resources. For more information, see Tagging Systems Manager documents (p. 1164).

11. In the **Step 1** section, provide the following information.

   - For **Step name**, enter a descriptive name for the first step of the automation workflow.
   - For **Action type**, select the action type to use for this step.
     
     For a list and information about the available action types, see Systems Manager Automation actions reference (p. 386).
   - For **Description**, enter a description for the automation step. You can use Markdown to format your text.
• Depending on the **Action type** selected, enter the required inputs for the action type in the **Step inputs** section. For example, if you selected the action `aws:approve`, you must specify a value for the `Approvers` property.

For information about the step input fields, see the entry in *Systems Manager Automation actions reference* (p. 386) for the action type you selected. For example: *aws:executeStateMachine – Run an AWS Step Functions state machine* (p. 420).

• (Optional) For **Additional inputs**, provide any additional input values needed for your document. The available input types depend on the action type you selected for the step. (Note that some action types require input values.)

  **Note**
  To add more inputs, choose **Add optional input**. To remove an input, choose the X (Remove) button.

• (Optional) For **Outputs**, enter any outputs for the execution of this step to make available for other processes.

  **Note**
  **Outputs** isn't available for all action types.

• (Optional) Expand the **Common properties** section and specify properties for the actions that are common to all Automation actions. For example, for **Timeout seconds**, you can provide a value in seconds to specify how long the step can run before it is stopped.

  For more information, see *Properties shared by all actions* (p. 387).

**Note**
To add more document steps, select **Add step** and repeat the procedure for creating a step. To remove a document step, choose **Remove step**.

12. Choose **Create automation** to save the document.

### Creating an Automation document using the Editor

If the AWS Systems Manager public Automation documents don't perform all the actions you want to perform on your AWS resources, you can create your own documents. For example, you can use the editor to modify parameters, add additional steps to an existing Automation document, or combine multiple Automation documents into a single document. If you're familiar with writing your own Automation documents in JSON or YAML, you can use the editor to enter the JSON or YAML document content.

For examples of custom Automation documents, see *Sample scenarios and custom Automation document solutions* (p. 539).

**Note**
If your Automation document uses the `aws:executeScript` Automation action with the `Attachment` input parameter, you must use the AWS CLI or Document Builder to successfully create the document.

The following procedure describes how to use the editor to create an Automation document.

**To create an Automation document using the editor**

2. In the navigation pane, choose **Documents**.

  -or-

---

357
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Documents in the navigation pane.

3. Choose Create automation.

4. For Name, type a descriptive name for the document.

5. Choose the Editor tab, and choose Edit.

6. Enter the document content using JSON or YAML.

7. Choose Create automation.

Creating Automation documents that run scripts

AWS Systems Manager Automation documents, or playbooks, support running scripts as part of the automation workflow. By using Automation documents, you can run scripts directly in AWS without creating a separate compute environment to run your scripts. Because Automation documents can run script steps along with other automation step types, such as approvals, you can manually intervene in critical or ambiguous situations.

Permissions for running Automation executions

To run an automation execution, Systems Manager must use the permissions of an AWS Identity and Access Management (IAM) role. The method that Automation uses to determine which role’s permissions to use depends on a few factors, and whether a step uses the aws:executeScript action.

For automation executions that do not use aws:executeScript, Automation uses one of two sources of permissions:

- The permissions of an IAM service role, or Assume role, that is specified in the Automation document or passed in as a parameter.
- If no IAM service role is specified, the permissions of the IAM user who started the automation execution.

When a step in an Automation document includes the aws:executeScript action, however, an IAM service role (Assume role) is always required if the Python or PowerShell script specified for the action is calling any AWS API actions. Automation checks for this role in the following order:

- The permissions of an IAM service role, or Assume role, that is specified in the Automation document or passed in as a parameter.
- A resource tag applied to the Automation document with the tag key AutomationScriptExecutionRole. In this case, Automation uses the IAM role that is specified as the tag value. For example, arn:aws:iam::123456789012:role/AutomationAssumeRole.

If no role is found, Automation attempts to run the Python or PowerShell script specified for aws:executeScript without any permissions. If the script is calling an AWS API operation (for example the Amazon EC2 CreateImage operation), or attempting to act on an AWS resource (such as an EC2 instance), the step containing the script fails, and Systems Manager returns an error message reporting the failure.

For more information about how to run an Automation workflow that uses an IAM service role or more advanced forms of delegated administration instead, see Running an Automation workflow by using an IAM service role (p. 337).

Adding scripts to Automation documents

You can add scripts to your Automation documents by including the script inline as part of a step in the document. You can also attach scripts to the document by uploading the scripts from your local machine...
or by specifying an Amazon Simple Storage Service (Amazon S3) bucket where the scripts are located. After a step that runs a script completes, the output of the script is available as a JSON object, which you can then use as input for subsequent steps in your Automation workflow.

**Script constraints for Automation documents**

The Automation action `aws:executeScript` currently supports running Python 3.6, Python 3.7, and PowerShell Core 6.0 scripts.

Automation documents enforce a limit of five file attachments. Scripts can either be in the form of a Python script (.py), a PowerShell Core script (.ps1), or attached as contents within a .zip file.

Your account is charged for running scripts using Automation. Automation steps that use the action `aws:executeScript` are considered special steps. There is no step limit for special steps, but your account is charged based on the number of steps and duration of your script execution. For more information, see the [AWS Systems Manager Pricing page](https://aws.amazon.com/pricing/).

The following topics describe how to create Automation documents that run scripts.

**Topics**
- Creating an Automation document that runs a script (console) (p. 359)
- Creating an Automation document that runs scripts (command line) (p. 361)
- Amazon managed Automation documents that run scripts (p. 364)

**Creating an Automation document that runs a script (console)**

**To create an Automation document that runs a script**

The following procedure describes how to use Document Builder in the Systems Manager console and to create a custom Automation document that runs a script that you provide.

2. In the navigation pane, choose **Documents**.
   - **or**
   
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Documents** in the navigation pane.
3. Choose **Create automation**.
4. For **Name**, type a descriptive name for the document.
5. For **Document description**, provide the markdown style description for the document. You can provide instructions for using the document, numbered steps, or any other type of information to describe the document. Refer to the default text for information about formatting your content.
   
   **Tip**
   
   Toggle between **Hide preview** and **Show preview** to see what your description content looks like as you compose.
6. (Optional) For **Assume role**, enter the name or ARN of a service role to perform actions on your behalf. If you don't specify a role, Automation uses the access permissions of the user who invokes the automation execution.

   **Important**

   For Automation documents not owned by Amazon that use the `aws:executeScript` action, a role must be specified. For information, see [Permissions for running Automation executions](https://docs.aws.amazon.com/systems-manager/latest/userguide/awsscm-permissions.html) (p. 358).
7. (Optional) For Outputs, enter any outputs for the execution of this document to make available for other processes.

   For example, if your document creates a new AMI, you might specify ["CreatImage.ImageId"], and then use this output to create new instances in a subsequent automation execution.

8. (Optional) Expand the Input parameters section and do the following.

   1. For Parameter name, enter a descriptive name for the document parameter you are creating.
   2. For Type, choose a type for the parameter, such as String or StringList.
   3. For Required, do one of the following.
      - Choose Yes if a value for this document parameter must be supplied at runtime.
      - Choose No if the parameter is not required, and (optional) enter a default parameter value in Default value.
   4. For Description, enter a description for the document parameter.

   **Note**
   To add more document parameters, choose Add a parameter. To remove a document parameter, choose the X (Remove) button.

9. (Optional) Expand the Target type sections and choose a target type to define the kinds of resources the document can run on. For example, to run a document on EC2 instances, choose /AWS::EC2::Instance.

   **Note**
   If you specify a value of '/', the document can run on all types of resources. For a list of valid resource types, see AWS Resource Types Reference in the AWS CloudFormation User Guide.

10. In the Step 1 section, provide the following information.

    - For Step name, enter a descriptive name for the first step of the automation workflow.
    - For Action type, select Run a script (aws:executeScript).
    - For Description, enter a description for the automation step. You can use Markdown to format your text.

11. Expand the Inputs section and provide the following information.

    - For Runtime, choose the type of script you are adding. Currently, Automation supports Python 3.6, Python 3.7, and PowerShell Core 6.0.
    - For Handler, enter the entry for script execution, usually the function name from your script. (Not required for PowerShell.)

        **Important**
        You must ensure the function defined in the handler has two parameters, events and context. For example, you would enter launch_instance if your script began with the following.

        ```python
        def launch_instance(events, context):
            import boto3
            ec2 = boto3.client('ec2')
            [...truncated...]
        ```

    - For Script, choose a method to provide a script to the Automation document.
      - To embed a script within the document, enter the script code in the text-box area.
      - Or-
      - For Attachment, choose either Stored on my machine or Upload S3 File URL.
If you choose **Stored on my machine**: For Amazon S3 URL, enter the location of an S3 bucket in your account where you want to store the upload attachment, and then choose **Upload** to browse to and select the file.

If you choose **Upload S3 File URL**, provide the following information:
- **S3 file url**: Enter the location in an S3 bucket in your account where the file is stored.
- **File name**: Enter the name of the file.
- **File checksum**: Enter the checksum of the file by using the sha256 algorithm.

**Tip**
You can calculate the checksum of the file in sha256 by using a tool like `shasum` in Linux. For example: `shasum -a 256 /path/to/file`. In Windows, you can use the `Get-FileHash` PowerShell cmdlet to obtain the same information. Note that the ETag or md5 checksum won't work for this value.

12. (Optional) Expand **Additional inputs** and do the following.
   - For **Input name**, chose **InputPayload** - Function input in YAML format.
   - For **Input value**, enter your script input in YAML format.

13. (Optional) Expand **Outputs** and enter the **Name**, **Selector**, and **Type** for any output to create from this step. Step output can be used in subsequent steps of your Automation document. Here are a few examples for demonstration.

   **Name**: `myInstance` | **Selector**: `$.InstanceInformationList[0].InstanceId` | **Type**: `String`

   **Name**: `platform` | **Selector**: `$.Reservations[0].Instances[0].Platform` | **Type**: `String`

   **Name**: `message` | **Selector**: `$.Payload.message` | **Type**: `String`

   For more information about outputs, see **Working with inputs and outputs (p. 378)**.

   **Tip**
   To add more outputs, select **Add output**.

14. (Optional) Expand the **Common properties** section and specify properties for the actions that are common to all Automation actions. For example, for **Timeout seconds**, you can provide a value in seconds to specify how long the step can run before it is stopped.

   For more information, see **Properties shared by all actions (p. 387)**.

   **Note**
   To add more document steps, select **Add step** and repeat the procedure for creating a step.
   To remove a document step, choose **Remove step**.

15. Choose **Create document** to save the document.

### Creating an Automation document that runs scripts (command line)

The following examples show how to use the AWS CLI (on Linux or Windows Server) or AWS Tools for PowerShell to create an Automation document that runs a script using the **Attachment** parameter.

**Before You Begin**

Before you begin, ensure you have the following resources prepared.

- The content for your Automation document in a YAML-formatted file with a `.yaml` extension, such as `MyAutomationDocument.yaml`.
- The files to attach to the document. You can attach script files or `.zip` files.

   For scripts, Automation supports Python 3.6 and 3.7, PowerShell Core 6.0.
• Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

For information, see Install or upgrade AWS command line tools (p. 57).

**Attach a single file from an S3 bucket**

Run the following command to create an Automation document using a script file stored in an S3 bucket.

**Linux**

```bash
aws ssm create-document \
  --name CustomAutomationScript \
  --content file://AutomationDocument.yaml \
  --document-format YAML \
  --attachments "Key=S3FileUrl,Name=script.py,Values=https://bucket-name.s3-aws-region.amazonaws.com/filePath" \
  --document-type Automation
```

**Windows**

```bash
aws ssm create-document ^
  --name CustomAutomationScript ^
  --content file://AutomationDocument.yaml ^
  --document-format YAML ^
  --attachments Key=S3FileUrl,Name=script.py,Values="https://bucket-name.s3-aws-region.amazonaws.com/filePath" ^
  --document-type Automation
```

**PowerShell**

```powershell
New-SSMDocument  
  -Name CustomAutomationScript  
  -Content file://AutomationDocument.yaml  
  -DocumentFormat YAML  
  -Attachments @{
    "Key"="S3FileUrl";
    "Name"="script.py";
    "Values"="https://bucket-name.s3-aws-region.amazonaws.com/filePath"
  }, @{
    "Key"="S3FileUrl";
    "Name"="Zip-file.zip";
    "Values"="https://bucket-name.s3-aws-region.amazonaws.com/filePath"
  }
  -DocumentType Automation
```

**Attach files from an S3 bucket**

Run the following command to create an Automation document using a script or multiple script files stored in an S3 bucket. Note that a Name key for files is not specified. The command attaches all supported files from the S3 bucket location.

**Linux**

```bash
aws ssm create-document \
  --name CustomAutomationScript \
  --content file://AutomationDocument.yaml \
  --document-format YAML \
  --attachments Key=SourceUrl,Values="https://bucket-name.s3-aws-region.amazonaws.com/filePath"
```
Attach files from another Automation document in your AWS account

Run the following command to create an Automation document using a script that is already attached to another Automation playbook in your account.

The format of the key value in this command is `document-name/document-version-number/file-name`. For example:

```
"MyDocument/2/script.py"
```

---

### Windows

```bash
aws ssm create-document ^
   --name CustomAutomationScript ^
   --content file://AutomationDocument.yaml ^
   --document-format YAML ^
   --attachments Key=SourceUrl,Values="https://bucket-name.s3-aws-region.amazonaws.com/filePath" ^
   --document-type Automation
```

### PowerShell

```powershell
New-SSMDocument `^`
   -Name CustomAutomationScript `^`
   -Content file://AutomationDocument.yaml `^`
   -DocumentFormat YAML `^`
   -Attachments @{
     "Key"="SourceUrl";
     "Values"="https://bucket-name.s3-aws-region.amazonaws.com/filePath"
   } `^`
   -DocumentType Automation
```

---

### Linux

```bash
aws ssm create-document \`
   --name CustomAutomationScript \`
   --content file://AutomationDocument.yaml \`
   --document-format YAML \`
   --attachments Key=AttachmentReference,Values="document-name/document-version-number/file-name" \`
   --document-type Automation
```

### Windows

```bash
aws ssm create-document ^
   --name CustomAutomationScript ^
   --content file://AutomationDocument.yaml ^
   --document-format YAML ^
   --attachments Key=AttachmentReference,Values="document-name/document-version-number/file-name" ^
   --document-type Automation
```

### PowerShell

```powershell
New-SSMDocument `^`
   -Name CustomAutomationScript `^`
   -Content file://AutomationDocument.yaml `^`
```
Attach files from an Automation document in another AWS account

Run the following command to create an Automation document using a script that is already attached to an Automation playbook that has been shared with you from another AWS account.

The format of the key value in this command is `document-arn/document-version-number/file-name`. For example:

```
```

Linux

```
aws ssm create-document \
     --name CustomAutomationScript \
     --content file://AutomationDocument.yaml \
     --document-format YAML \
     --attachments Key=AttachmentReference,Values="document-arn/document-version-number/file-name"/file-name" \
     --document-type Automation
```

Windows

```
aws ssm create-document ^
     --name CustomAutomationScript ^
     --content file://AutomationDocument.yaml ^
     --document-format YAML ^
     --attachments Key=AttachmentReference,Values="document-arn/document-version-number/file-name"^/file-name" ^
     --document-type Automation
```

PowerShell

```
New-SSMDocument `\n     -Name CustomAutomationScript `\n     -Content file://AutomationDocument.yaml `\n     -DocumentFormat YAML `\n     -Attachments @{ `\n         "Key"="AttachmentReference"; `\n         "Values"="document-arn/document-version-number/file-name" `\n     } `\n     -DocumentType Automation
```

Amazon managed Automation documents that run scripts

AWS Systems Manager Automation documents, or playbooks, support running scripts as part of the automation execution.

The following are AWS managed Automation documents that include support for running scripts.

**Note**

You can also create your own custom Automation documents that can run scripts. For information, see [*Creating Automation documents that run scripts*](p. 358).
Creating dynamic Automation workflows with conditional branching

By default, the steps that you define in the `mainSteps` section of an Automation document, or playbook, run in sequential order. After one action is completed, the next action specified in the `mainSteps` section begins. Furthermore, if an action fails to run, the entire Automation workflow fails (by default). You can use the `aws:branch` Automation action and the Automation document options described in this section to create Automation workflows that perform conditional branching. This means that you can create Automation workflows that jump to a different step after evaluating different choices or that dynamically respond to changes when a step completes. Here is a list of options that you can use to create dynamic Automation workflows:

- **aws:branch**: This automation action enables you to create a dynamic Automation workflow that evaluates multiple choices in a single step and then jumps to a different step in the Automation document based on the results of that evaluation.
- **nextStep**: This option specifies which step in an Automation workflow to process next after successfully completing a step.
- **isEnd**: This option stops an Automation execution at the end of a specific step. The default value for this option is false.
- **isCritical**: This option designates a step as critical for the successful completion of the Automation. If a step with this designation fails, then Automation reports the final status of the Automation as Failed. The default value for this option is true.
- **onFailure**: This option indicates whether the workflow should abort, continue, or go to a different step on failure. The default value for this option is abort.

The following section describes the `aws:branch` Automation action. For more information about the nextStep, isEnd, isCritical, and onFailure workflow options, see Examples of how to use dynamic workflow options (p. 373).

Working with the `aws:branch` action

The `aws:branch` action offers the most dynamic conditional branching options for Automation workflows. As noted earlier, this action enables your Automation workflow to evaluate multiple conditions in a single step and then jump to a new step based on the results of that evaluation. The `aws:branch` action functions like an IF-ELIF-ELSE statement in programming.

Here is a YAML example of an `aws:branch` step:

```yaml
- name: ChooseOSforCommands
  action: aws:branch
  inputs:
```
When you specify the `aws:branch` action for a step, you specify Choices that the workflow must evaluate. The workflow can evaluate Choices based on the value of a parameter that you specified in the Parameters section of the Automation document. The workflow can also evaluate Choices based on output from a previous step.

The Automation workflow evaluates each choice by using a Boolean expression. If the evaluation determines that the first choice is true, then the workflow jumps to the step designated for that choice. If the evaluation determines that the first choice is false, then the workflow evaluates the next choice. If your step includes three or more Choices, then the workflow evaluates each choice in sequential order until it evaluates a choice that is true. The workflow then jumps to the designated step for the true choice.

If none of the Choices are true, the workflow checks to see if the step contains a Default value. A Default value defines a step that the workflow should jump to if none of the choices are true. If no Default value is specified for the step, then the Automation workflow processes the next step in the document.

Here is an `aws:branch` step in YAML named `chooseOSfromParameter`. The step includes two Choices: (NextStep: runWindowsCommand) and (NextStep: runLinuxCommand). The Automation workflow evaluates these Choices to determine which command to run for the appropriate operating system. The Variable for each choice uses `{{OSName}}`, which is a parameter that the document author defined in the Parameters section of the document.

Here is an `aws:branch` step in YAML named `chooseOSfromOutput`. The step includes two Choices: (NextStep: runPowerShellCommand) and (NextStep: runShellCommand). The Automation workflow evaluates these Choices to determine which command to run for the appropriate operating system. The Variable for each choice uses `{{GetInstance.platform}}`, which is the output from an earlier step in the document. This example also includes an option called Default. If the workflow evaluates both Choices, and neither choice is true, then the Automation workflow jumps to a step called PostProcessing.
Creating an `aws:branch` step in an Automation document

When you create an `aws:branch` step in an Automation document, you define the Choices the workflow should evaluate to determine which step the workflow should jump to next. As noted earlier, Choices are evaluated by using a Boolean expression. Each choice must define the following options:

- **NextStep**: The next step in the Automation document to process if the designated choice is true.
- **Variable**: Specify either the name of a parameter that is defined in the Parameters section of the Automation document, or specify an output object from a previous step in the Automation document.

Specify parameter variables by using the following form:

Variable: "{{name_of_parameter}}"

Specify output object variables by using the following form:

Variable: "{{previousStepName.outputFieldName}}"

**Note**  
Creating the output variable is described in more detail in the next section, About creating the output variable (p. 368).

- **Operation**: The criteria used to evaluate the choice, such as `StringEquals: Linux`. The `aws:branch` action supports the following operations:

  **String operations**
  - `StringEquals`
  - `EqualsIgnoreCase`
  - `StartsWith`
  - `EndsWith`
  - `Contains`

  **Numeric operations**
  - `NumericEquals`
  - `NumericGreater`
  - `NumericLesser`
  - `NumericGreaterOrEquals`
  - `NumericLesser`
  - `NumericLesserOrEquals`

  **Boolean operation**
  - `BooleanEquals`

  **Important**  
  When you create an Automation document, the system validates each operation in the document. If an operation is not supported, the system returns an error when you try to create the document.

- **Default**: Specify a fallback step that the workflow should jump to if none of the Choices are true.
Note
If you don't want to specify a Default value, then you can specify the isEnd workflow option. If none of the Choices are true and no Default value is specified, then the Automation workflow stops at the end of the step.

Use the following templates to help you construct the aws:branch step in your Automation documents:

YAML

```yaml
mainSteps:
  - name: a name for the step
    action: aws:branch
    inputs:
      Choices:
        - NextStep: step to jump to if evaluation for this choice is true
          Variable: "{{parameter name or output from previous step}}"
          Operation type: Operation value
        - NextStep: step to jump to if evaluation = true
          Variable: "{{parameter name or output from previous step}}"
          Operation type: Operation value
      Default:
        step to jump to if all choices are false
```

JSON

```json
{
  "mainSteps": [  
    {  
      "name": "a name for the step",
      "action": "aws:branch",
      "inputs": {  
        "Choices": [  
          {  
            "NextStep": "step to jump to if evaluation for this choice is true",
            "Variable": "{{parameter name or output from previous step}}",
            "Operation type": "Operation value"
          },  
          {  
            "NextStep": "step to jump to if evaluation = true",
            "Variable": "{{parameter name or output from previous step}}",
            "Operation type": "Operation value"
          }
        ],  
        "Default": "step to jump to if all choices are false"
      }
    }
  ]
}
```

About creating the output variable

To create an aws:branch choice that references the output from a previous step, you need to identify the name of the previous step and the name of the output field. You then combine the names of the step and the field by using the following format:

Variable: "{{previousStepName.outputFieldName}}"

For example, the first step below is named GetInstance. And then, under outputs, there is a field called platform. In the second step (ChooseOSforCommands), the author wants to reference the output from the platform field as a variable. To create the variable, simply
combine the step name (GetInstance) and the output field name (platform) to create Variable: "{{GetInstance.platform}}".

```json
mainSteps:
  - Name: GetInstance
    action: aws:executeAwsApi
    inputs:
      Service: ssm
      Api: DescribeInstanceInformation
      Filters:
        - Key: InstanceIds
          Values: ["{{ InstanceId }}"]
    outputs:
      - Name: myInstance
        Selector: ".InstanceInformationList[0].InstanceId"
        Type: String
      - Name: platform
        Selector: ".InstanceInformationList[0].PlatformType"
        Type: String
      - name: ChooseOSforCommands
        action: aws:branch
        inputs:
          Choices:
            - NextStep: runPowerShellCommand
              Variable: "{{GetInstance.platform}}"
              StringEquals: Windows
            - NextStep: runShellCommand
              Variable: "{{GetInstance.platform}}"
              StringEquals: Linux
            Default:
              Sleep
```

Here is a JSON example that shows how "Variable": "{{ describeInstance.Platform }}" is created from the previous step ("describeInstance") and the output field ("Platform").

```json
{
  "name": "describeInstance",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "DescribeInstances",
    "InstanceIds": [
      "{{ InstanceId }}"
    ]
  },
  "outputs": [
    {
      "Name": "Platform",
      "Selector": ".Reservations[0].Instances[0].Platform",
      "Type": "String"
    }
  ],
  "nextStep": "branchOnInstancePlatform"
},
{
  "name": "branchOnInstancePlatform",
  "action": "aws:branch",
  "inputs": {
    "Choices": [
      {
        "NextStep": "runEC2RescueForWindows",
        "Variable": "{{ describeInstance.Platform }}",
        "StringEquals": "windows"
      }
    ]
  }
}
```
Example aws:branch Automation documents

Here are some example Automation documents that use aws:branch.

**Example 1: Using aws:branch with an output variable to run commands based on the operating system type**

In the first step of this sample (GetInstance), the document author uses the aws:executeAwsApi action to call the ssm DescribeInstanceInformation API action. The author uses this action to determine the type of operating system being used by an instance. The aws:executeAwsApi action outputs the instance ID and the platform type.

In the second step (ChooseOSforCommands), the author uses the aws:branch action with two Choices (NextStep: runPowerShellCommand) and (NextStep: runShellCommand). The Automation workflow evaluates the operating system of the instance by using the output from the previous step (Variable: "{{GetInstance.platform}}"). The Automation workflow jumps to a step for the designated operating system.

```json
---
schemaVersion: '0.3'
assumeRole: "{{AutomationAssumeRole}}"
parameters:
  AutomationAssumeRole:
    default: ""
    type: String
mainSteps:
  - name: GetInstance
    action: aws:executeAwsApi
    inputs:
      Service: ssm
      Api: DescribeInstanceInformation
    outputs:
      - Name: myInstance
        Selector: "$.InstanceInformationList[0].InstanceId"
        Type: String
      - Name: platform
        Selector: "$.InstanceInformationList[0].PlatformType"
        Type: String
  - name: ChooseOSforCommands
    action: aws:branch
    inputs:
      Choices:
      - NextStep: runPowerShellCommand
        Variable: "{{GetInstance.platform}}"
        StringEquals: Windows
      - NextStep: runShellCommand
        Variable: "{{GetInstance.platform}}"
        StringEquals: Linux
        Default:
        Sleep
  - name: runShellCommand
    action: aws:runCommand
    inputs:
      DocumentName: AWS-RunShellScript
      InstanceIds:
      - "{{GetInstance.myInstance}}"
      Parameters:
```
Example 2: Using aws:branch with a parameter variable to run commands based on the operating system type

The document author defines several parameter options at the beginning of the document in the parameters section. One parameter is named OperatingSystemName. In the first step (ChooseOS), the author uses the aws:branch action with two Choices (NextStep: runWindowsCommand) and (NextStep: runLinuxCommand). The variable for these Choices references the parameter option specified in the parameters section (Variable: "{{OperatingSystemName}}"). When the user runs this Automation workflow, they specify a value at runtime for OperatingSystemName. The Automation workflow uses the runtime parameter during the Choices evaluation. The Automation workflow jumps to a step for the designated operating system based on the runtime parameter specified for OperatingSystemName.
Creating complex branching documents with operators

You can create complex branching documents by using the And, Or, and Not operators in your aws:branch steps.

The 'And' Operator

Use the And operator when you want multiple variables to be true for a choice. In the following example, the first choice evaluates if an instance is running and uses the Windows operating system. If the evaluation of both of these variables is true, then the Automation workflow jumps to the runPowerShellCommand step. If one or more of the variables is false, then the workflow evaluates the variables for the second choice.

The 'Or' Operator

Use the Or operator when you want any of multiple variables to be true for a choice. In the following example, the first choice evaluates if a parameter string is Windows and if the output from an AWS Lambda step is true. If the evaluation determines that either of these variables is true, then the Automation workflow jumps to the RunPowerShellCommand step. If both variables are false, then the workflow evaluates the variables for the second choice.
The 'Not' Operator

Use the Not operator when you want to jump to a step defined when a variable is not true. In the following example, the first choice evaluates if a parameter string is Not Linux. If the evaluation determines that the variable is not Linux, then the Automation workflow moves to the sleep2 step. If the evaluation of the first choice determines that it is Linux, then the workflow evaluates the next choice.

```json
mainSteps:
  - name: switch
    action: aws:branch
    inputs:
      Choices:
      - NextStep: sleep2
        Not:
          Variable: "{{testParam}}"
          StringEquals: Linux
      - NextStep: sleep1
        Variable: "{{testParam}}"
        StringEquals: Windows
      Default:
        sleep3
```

Examples of how to use dynamic workflow options

This section includes different examples of how to use dynamic workflow options in an Automation document. Each example in this section extends the following Automation document. This document has two actions. The first action is named InstallMsiPackage. It uses the aws:runCommand action to install an application on a Windows Server instance. The second action is named TestInstall. It uses the aws:invokeLambdaFunction action to perform a test of the installed application if the application installed successfully. Step one specifies onFailure: Abort. This means that if the application did not install successfully, the Automation workflow execution stops before step two.

Example 1: Automation document with two linear actions

```json
---
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:
```

```json
- Variable: "{{parameter1}}"
  StringEquals: Windows
- Variable: "{{BooleanParam1}}"
  BooleanEquals: true
  NextStep: RunPowershellCommand
- Or:
  - Variable: "{{parameter2}}"
    StringEquals: Linux
  - Variable: "{{BooleanParam2}}"
    BooleanEquals: true
    NextStep: RunShellScript
```
Creating a Dynamic Workflow that Jumps to Different Steps by Using the onFailure Option

The following example uses the onFailure: step:step_name, nextStep, and isEnd options to create a dynamic Automation workflow. With this example, if the InstallMsiPackage action fails, then the workflow jumps to an action called PostFailure (onFailure: step:PostFailure) to run an AWS Lambda function to perform some action in the event the install failed. If the install succeeds, then the workflow process jumps to the TestInstall action (nextStep: TestInstall). Both the TestInstall and the PostFailure steps use the isEnd option (isEnd: true) so that the workflow finishes the workflow execution when either of those steps is completed.

**Note**
Using the isEnd option in the last step of the mainSteps section is optional. If the last step does not jump to other steps, then the Automation workflow stops after running the action in the last step.

Example 2: A dynamic workflow that jumps to different steps

```plaintext
mainSteps
- name: InstallMsiPackage
  action: aws:runCommand
  onFailure: step:PostFailure
  maxAttempts: 2
  inputs:
    InstanceIds:
    - i-02573cafefEXAMPLE
    - i-0471e04240EXAMPLE
    - i-07782c72fEXAMPLE
  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
```

374
Note
Before processing an Automation document, the system verifies that the document does not create an infinite loop. If an infinite loop is detected, Automation returns an error and a circle trace showing which steps create the loop.

Creating a Dynamic Workflow that Defines Critical Steps
You can specify that a step is critical for the overall success of the Automation workflow. If a critical step fails, then Automation reports the status of the execution as failed, even if one or more steps ran successfully. In the following example, the user identifies the `VerifyDependencies` step if the `InstallMsiPackage` step fails (`onFailure: step:VerifyDependencies`). The user specifies that the `InstallMsiPackage` step is not critical (`isCritical: false`). In this example, if the application failed to install, Automation processes the `VerifyDependencies` step to determine if one or more dependencies is missing, which therefore caused the application install to fail.

Example 3: Defining critical steps for the Automation workflow

```yaml
---
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
```
action: 'aws:waitForAwsResourceProperty'
timeoutSeconds: 600
isCritical: false
onFailure: 'step:getCurrentImageState'
inputs:
  Service: ec2
  Api: DescribeImages
  ImageIds:
    - '{{ createImage.newImageId }}'
  PropertySelector: '$.Images[0].State'
  DesiredValues:
    - available
nextStep: copyImage

JSON

```
{
  "name": "verifyImageAvailability",
  "action": "aws:waitForAwsResourceProperty",
  "timeoutSeconds": 600,
  "isCritical": false,
  "onFailure": "step:getCurrentImageState",
  "inputs": {
    "Service": "ec2",
    "Api": "DescribeImages",
    "ImageIds": [
      "{{ createImage.newImageId }}"
    ],
    "PropertySelector": "$.Images[0].State",
    "DesiredValues": [
      "available"
    ]
  },
  "nextStep": "copyImage"
}
```

For more information about properties shared by all Automation actions, see Properties shared by all actions (p. 387).

Invoking other AWS services from a Systems Manager Automation workflow

You can invoke other AWS services and other Systems Manager capabilities in your Automation workflow by using the following Automation actions in your Automation documents, or playbooks.

- **aws:executeAwsApi**: This Automation action calls and runs AWS API actions. Most API actions are supported, although not all API actions have been tested. For example, the following API actions are supported: CreateImage, Delete bucket, RebootDBInstance, and CreateGroups, to name a few. Streaming API actions, such as the Get Object action, aren’t supported.

- **aws:waitForAwsResourceProperty**: This Automation action enables your workflow to wait for a specific resource state or event state before continuing the workflow. For example, you can use this action with the Amazon Relational Database Service (Amazon RDS) DescribeDBInstances API action to pause an Automation workflow so that a database instance has time to start.

- **aws:assertAwsResourceProperty**: This Automation action enables you to assert a specific resource state or event state for a specific Automation step. For example, you can specify that an Automation step must wait for an EC2 instance to start. Then it will call the Amazon EC2 DescribeInstanceStatus API action with the DesiredValue property of running. This ensures that the Automation workflow waits for a running instance and then continues when the instance is, in fact, running.
Here is a sample Automation document in YAML that uses the aws:executeAwsApi action to disable read and write permissions on an S3 bucket.

```yaml
---
description: Disable S3-Bucket's public WriteRead access via private ACL
schemaVersion: "0.3"
assumeRole: "{{ AutomationAssumeRole }}"
parameters:
  S3BucketName:
    type: String
    description: (Required) S3 Bucket subject to access restriction
  AutomationAssumeRole:
    type: String
    description: (Optional) The ARN of the role that allows Automation to perform the
actions on your behalf.
    default: ""
mainSteps:
  - name: DisableS3BucketPublicReadWrite
    action: aws:executeAwsApi
    inputs:
      Service: s3
      Api: PutBucketAcl
      Bucket: "{{S3BucketName}}"
      ACL: private
      isEnd: true
---
```

Here is a sample Automation document in YAML that uses all three actions. The document does the following:

- Uses the aws:executeAwsApi action to call the Amazon EC2 DescribeImages API action to get the name of a specific Windows Server 2016 AMI. It outputs the image ID as ImageId.
- Uses the aws:executeAwsApi action to call the Amazon EC2 RunInstances API action to launch one instance that uses the ImageId from the previous step. It outputs the instance ID asInstanceId.
- Uses the aws:waitForAwsResourceProperty action to poll the Amazon EC2 DescribeInstanceStatus API action to wait for the instance to reach the running state. The action times out in 60 seconds. The step times out if the instance state failed to reach running after 60 seconds of polling.
- Uses the aws:assertAwsResourceProperty action to call the Amazon EC2 DescribeInstanceStatus API action to assert that the instance is in the running state. The step fails if the instance state is not running.

```yaml
---
description: Sample Automation Document Using AWS API Actions
schemaVersion: '0.3'
assumeRole: "{{ AutomationAssumeRole }}"
parameters:
  AutomationAssumeRole:
    type: String
    description: "(Optional) The ARN of the role that allows Automation to perform the
actions on your behalf."
    default: ""
  ImageName:
    type: String
    description: "(Optional) Image Name to launch EC2 instance with."
    default: "Windows_Server-2016-English-Full-Base-2018.07.11"
mainSteps:
  - name: getImageId
    action: aws:executeAwsApi
    inputs:
      Service: ec2
```
Working with inputs and outputs

Each of the previously described Automation actions enables you to call a specific API action by specifying the service namespace, the API action name, the input parameters, and the output parameters. Inputs are defined by the API action that you choose. You can view the API actions (also called methods) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all API actions (methods) for Amazon Relational Database Service (Amazon RDS) are listed on the following page: Amazon RDS methods.

You can view the schema for each Automation action in the following locations:

- `aws:assertAwsResourceProperty` – Assert an AWS resource state or event state (p. 395)
- `aws:executeAwsApi` – Call and run AWS API actions (p. 415)
- `aws:waitForAwsResourceProperty` – Wait on an AWS resource property (p. 432)
The schemas include descriptions of the required fields for using each action.

**Using the Selector/PropertySelector Fields**

Each Automation action requires that you specify either an output Selector (for `aws:executeAwsApi`) or a PropertySelector (for `aws:assertAwsResourceProperty` and `aws:waitForAwsResourceProperty`). These fields are used to process the JSON response from an AWS API action. These fields use the JSONPath syntax.

Here is an example to help illustrate this concept for the `aws:executeAwsApi` action:

```yaml
---
mainSteps:
- name: getImageId
  action: aws:executeAwsApi
  inputs:
    Service: ec2
    Api: DescribeImages
    Filters:
      - Name: "name"
        Values:
          - "{{ ImageName }}"
  outputs:
    - Name: ImageId
      Selector: "$.Images[0].ImageId"
      Type: "String"
...```

In the `aws:executeAwsApi` step `getImageId`, the workflow invokes the `DescribeImages` API action and receives a response from `ec2`. The workflow then applies `Selector - "$.Images[0].ImageId"` to the API response and assigns the selected value to the output `ImageId` variable. Other steps in the same Automation workflow can use the value of `ImageId` by specifying `"{{ getImageId.ImageId }}"`.

Here is an example to help illustrate this concept for the `aws:waitForAwsResourceProperty` action:

```yaml
---
- name: waitUntilInstanceStateRunning
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 60
  inputs:
    Service: ec2
    Api: DescribeInstanceStatus
    InstanceIds:
      - "{{ launchOneInstance.InstanceId }}"
    PropertySelector: "$.InstanceStatuses[0].InstanceState.Name"
    DesiredValues:
      - running
...```

In the `aws:waitForAwsResourceProperty` step `waitUntilInstanceStateRunning`, the workflow invokes the `DescribeInstanceStatus` API action and receives a response from `ec2`. The workflow then applies `PropertySelector - "$.InstanceStatuses[0].InstanceState.Name"` to the response and checks if the specified returned value matches a value in the `DesiredValues` list (in this case `running`). The step repeats the process until the response returns an instance state of `running`.

**Using JSONPath in an Automation workflow**

A JSONPath expression is a string beginning with "$." that is used to select one of more components within a JSON element. The following list includes information about JSONPath operators that are supported by Systems Manager Automation:
- **Dot-notated child (.)**: Use with a JSON object. This operator selects the value of a specific key.
- **Deep-scan (..)**: Use with a JSON element. This operator scans the JSON element level by level and selects a list of values with the specific key. Note that the return type of this operator is always a JSON array. In the context of an Automation step output type, the operator can be either StringList or MapList.
- **Array-Index ([ ])**: Use with a JSON array. This operator gets the value of a specific index.

To better understand JSONPath operators, review the following JSON response from the ec2 DescribeInstances API action. Below this response are several examples that show different results by applying different JSONPath expressions to the response from the DescribeInstances API action.

```json
{
    "NextToken": "abcdefg",
    "Reservations": [
        {
            "OwnerId": "123456789012",
            "ReservationId": "r-abcd12345678910",
            "Instances": [
                {
                    "InstanceId": "ami-12345678",
                    "BlockDeviceMappings": [
                        {
                            "Ebs": {
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-000000000000"
                            },
                            "DeviceName": "/dev/xvda"
                        }
                    ],
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    }
                }
            ],
            "Groups": []
        },
        {
            "OwnerId": "123456789012",
            "ReservationId": "r-12345678910abcd",
            "Instances": [
                {
                    "InstanceId": "ami-12345678",
                    "BlockDeviceMappings": [
                        {
                            "Ebs": {
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-1111111111"
                            },
                            "DeviceName": "/dev/xvda"
                        }
                    ],
                    "State": {
                        "Code": 80,
                        "Name": "stopped"
                    }
                }
            ],
            "Groups": []
        }
    ]
}
```
JSONPath Example 1: Get a specific String from a JSON response

JSONPath:
$.
Returns:
"ami-12345678"
Type: String

JSONPath Example 2: Get a specific Boolean from a JSON response

JSONPath:
$.
Returns:
true
Type: Boolean

JSONPath Example 3: Get a specific Integer from a JSON response

JSONPath:
$.
Returns:
16
Type: Integer

JSONPath Example 4: Deep scan a JSON response, then get all of the values for VolumeId as a StringList

JSONPath:
$.
Returns:
[
  "vol-000000000000",
  "vol-111111111111"
]
Type: StringList

JSONPath Example 5: Get a specific BlockDeviceMappings object as a StringMap

JSONPath:
$.
Returns:
{
  "Ebs" : {
    "DeleteOnTermination" : true,
    "Status" : "attached",
    "VolumeId" : "vol-000000000000"
  },
  "DeviceName" : "/dev/xvda"
Type: StringMap

JSONPath Example 6: Deep scan a JSON response, then get all of the State objects as a MapList

JSONPath:
$.Reservations..Instances..State

Returns:
[
  {
    "Code" : 16,
    "Name" : "running"
  },
  {
    "Code" : 80,
    "Name" : "stopped"
  }
]

Type: MapList

Important
If you run an automation workflow that invokes other services by using an AWS Identity and Access Management (IAM) service role, be aware that the service role must be configured with permission to invoke those services. This requirement applies to all AWS Automation documents (AWS-* documents) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RestartEC2Instance documents, to name a few. This requirement also applies to any custom Automation documents you create that invoke other AWS services by using actions that call other services. For example, if you use the aws:executeAwsApi, aws:createStack, or aws:copyImage actions, then you must configure the service role with permission to invoke those services. You can enable permissions to other AWS services by adding an IAM inline policy to the role. For more information, see (Optional) add an Automation inline policy to invoke other AWS services (p. 284).

Sample walkthrough: Start an Amazon RDS instance from a Systems Manager Automation workflow

This sample walkthrough shows you how to create and run an Automation document in YAML that uses all three API actions to see if an Amazon Relational Database Service (Amazon RDS) database instance is running. If the instance isn't running, the workflow starts it.

To invoke an Amazon RDS API action from a Systems Manager Automation

1. Open a text editor and paste the following Automation document content into the file. Specify an Automation role and the instance ID to check. You will add the mainSteps actions later.

```yaml
---
description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "The_Automation_role_to_use_when_running_the_document"
parameters:
  InstanceId: The_instance_ID_to_start
    type: String
    description: (Required) RDS instance ID to start
  AutomationAssumeRole:
    type: String
    description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
```
2. For the first step of the workflow, you need to determine if the instance is already running. You can use the aws:assertAwsResourceProperty action to determine and assert a specific instance status. Before you can add the aws:assertAwsResourceProperty action to the document, you must determine and specify the required inputs. The following list describes how to determine and specify the required inputs. You can view an example of how to enter this information in the Automation document following the list.

a. View the schema to see all available inputs for the `aws:assertAwsResourceProperty` action.

b. Determine the namespace of the service to invoke. You can view a list of AWS service namespaces in Amazon Resource Names (ARNs) and AWS Service Namespaces in the Amazon Web Services General Reference. The namespace for Amazon RDS is `rds`.

c. Determine which Amazon RDS API action enables you to view the status of a database instance. You can view the API actions (also called methods) on the Amazon RDS methods page.

d. Specify one or more request parameters for the DescribeDBInstances API action. For example, this action uses the `DBInstanceIdentifier` request parameter.

e. Determine one or more PropertySelectors. A PropertySelector is a response object that is returned by the request of this API action. For example, on the Amazon RDS methods, choose the `describe_db_instances` method and scroll down to the Response Structure section. `DBInstances` is listed as a response object. For the purposes of this walkthrough, specify `DBInstances` and `DBInstanceStatus` as the PropertySelectors. Remember that PropertySelectors are entered by using JSONPath. This means that you format the information in the Automation document like this:

```
PropertySelector: "$.DBInstances[0].DBInstanceStatus".
```

f. Specify one or more DesiredValues. If you don't know the values you want to specify, then run the DescribeDBInstances API action to determine possible values. For this walkthrough, specify `available` and `starting`.

g. Enter the information you collected into the Automation document as shown in the following example.

```yaml
---
description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "The_Automation_role_to_use_when_running_the_document"
parameters:
  InstanceId: The_instance_ID_to_start
type: String
description: (Required) RDS Instance Id to stop
AutomationAssumeRole:
type: String
description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
default: ""
mainSteps:
  - name: AssertNotStartingOrAvailable
    action: aws:assertAwsResourceProperty
    isCritical: false
    onFailure: step:StartInstance
    nextStep: CheckStart
    inputs:
      Service: rds
      Api: DescribeDBInstances
      DBInstanceIdentifier: "{{InstanceId}}"
```
3. Specify an `aws:executeAwsApi` action in the `mainSteps` section to start the instance if the previous action determined that it is not started.

   a. View the schema to see all available inputs for `aws:executeAwsApi` – Call and run AWS API actions (p. 415).

   b. Specify the Amazon RDS `StartDBInstance` API action to start the instance.

   c. Enter the information you collected into the Automation document as shown in the following example.

```yaml
---
description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "{{ The_Automation_role_to_use_when_running_the_document }}"
parameters:
  InstanceId:
    type: String
    description: (Required) RDS Instance Id to stop
  AutomationAssumeRole:
    type: String
    description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
    default: ""
mainSteps:
  - name: AssertNotStartingOrAvailable
    action: aws:assertAwsResourceProperty
    isCritical: false
    onFailure: step:StartInstance
    nextStep: CheckStart
    inputs:
      Service: rds
      Api: DescribeDBInstances
      DBInstanceIdentifier: "{{InstanceId}}"
      PropertySelector: "$.DBInstances[0].DBInstanceStatus"
      DesiredValues: ["available", "starting"]
  - name: StartInstance
    action: aws:executeAwsApi
    inputs:
      Service: rds
      Api: StartDBInstance
      DBInstanceIdentifier: "{{InstanceId}}"
---
```

4. Specify an `aws:waitForAwsResourceProperty` action in the `mainSteps` section to wait for the instance to start before finishing the Automation workflow.

   a. View the schema to see all available inputs for the `aws:waitForAwsResourceProperty` – Wait on an AWS resource property (p. 432).

   b. Specify the Amazon RDS `DescribeDBInstances` API action to determine the instance status.

   c. Specify `$.DBInstances[0].DBInstanceStatus` as the `PropertySelector`

   d. Specify `available` as the `DesiredValue`.

   e. Enter the information you collected into the Automation document as shown in the following example.

```yaml
---
```

384
description: Start RDS instance
schemaVersion: "0.3"
assumeRole: "{{ The_Automation_role_to_use_when_running_the_document }}"

parameters:
  InstanceId:
    type: String
    description: (Required) RDS Instance Id to stop
  AutomationAssumeRole:
    type: String
    description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
    default: ""

mainSteps:
  - name: AssertNotStartingOrAvailable
    action: aws:assertAwsResourceProperty
    isCritical: false
    onFailure: step:StartInstance
    nextStep: CheckStart
    inputs:
      Service: rds
      Api: DescribeDBInstances
      DBInstanceIdentifier: "{{InstanceId}}"
      PropertySelector: "$.DBInstances[0].DBInstanceStatus"
      DesiredValues: ["available", "starting"]
  - name: StartInstance
    action: aws:executeAwsApi
    inputs:
      Service: rds
      Api: StartDBInstance
      DBInstanceIdentifier: "{{InstanceId}}"
  - name: CheckStart
    action: aws:waitForAwsResourceProperty
    onFailure: Abort
    maxAttempts: 10
    timeoutSeconds: 600
    inputs:
      Service: rds
      Api: DescribeDBInstances
      DBInstanceIdentifier: "{{InstanceId}}"
      PropertySelector: "$.DBInstances[0].DBInstanceStatus"
      DesiredValues: ["available"]
    isEnd: true

5. Save the file as sample.yaml.
6. Run the following command in the AWS CLI to add the document to your AWS account.

```bash
aws ssm create-document --name sampleDoc --document-type Automation --document-format YAML --content file://sample.yaml
```
7. Run the following command to run the Automation execution by using the document you just created. Make a note of the execution ID returned by Systems Manager after you start the execution.

```bash
aws ssm start-automation-execution --document-name sampleDoc
```
8. Run the following command to view the execution status.

```bash
aws ssm get-automation-execution --automation-execution-id automation_execution_id
```
Predefined Automation documents that invoke AWS APIs

Systems Manager Automation includes the following predefined SSM Automation documents that invoke AWS APIs.

<table>
<thead>
<tr>
<th>Document name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS-StartRdsInstance</td>
<td>Start an Amazon RDS instance.</td>
</tr>
<tr>
<td>AWS-StopRdsInstance</td>
<td>Stop an Amazon RDS instance.</td>
</tr>
<tr>
<td>AWS-RebootRdsInstance</td>
<td>Reboot an Amazon RDS instance.</td>
</tr>
<tr>
<td>AWS-CreateSnapshot</td>
<td>Create an Amazon Elastic Block Store (Amazon EBS) volume snapshot.</td>
</tr>
<tr>
<td>AWS-DeleteSnapshot</td>
<td>Delete an Amazon EBS volume snapshot.</td>
</tr>
<tr>
<td>AWS-ConfigureS3BucketLogging</td>
<td>Enable logging on an Amazon Simple Storage Service (Amazon S3) bucket.</td>
</tr>
<tr>
<td>AWS-DisableS3BucketPublicReadWrite</td>
<td>Disable read and write permissions on an S3 bucket by using a private ACL.</td>
</tr>
<tr>
<td>AWS-ConfigureS3BucketVersioning</td>
<td>Enable or suspend versioning on an S3 bucket.</td>
</tr>
<tr>
<td>AWS-DeleteDynamoDbBackup</td>
<td>Delete a Amazon DynamoDB (DynamoDB) table backup.</td>
</tr>
</tbody>
</table>

Either click the links in the table above, or use the following procedure to view more details about these Automation documents in the Systems Manager console.

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Documents in the navigation pane.
3. Choose a document, and then choose View details.
4. Choose the Content tab.

Systems Manager Automation actions reference

This reference describes the Automation actions that you can specify in an AWS Systems Manager Automation document. These actions cannot be used in other types of SSM documents. For information about plugins for other types of SSM documents, see SSM document plugin reference (p. 1052).

Systems Manager Automation runs steps defined in Automation documents. Each step is associated with a particular action. The action determines the inputs, behavior, and outputs of the step. Steps are defined in the mainSteps section of your Automation document.

You don't need to specify the outputs of an action or step. The outputs are predetermined by the action associated with the step. When you specify step inputs in your Automation documents, you can reference one or more outputs from an earlier step. For example, you can make the output of aws:runInstances available for a subsequent aws:runCommand action. You can also reference outputs from earlier steps in the Output section of the Automation document.
Important
If you run an automation workflow that invokes other services by using an AWS Identity and Access Management (IAM) service role, be aware that the service role must be configured with permission to invoke those services. This requirement applies to all AWS Automation documents (AWS-* documents) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RestartEC2Instance documents, to name a few. This requirement also applies to any custom Automation documents you create that invoke other AWS services by using actions that call other services. For example, if you use the aws:executeAwsApi, aws:createStack, or aws:copyImage actions, then you must configure the service role with permission to invoke those services. You can enable permissions to other AWS services by adding an IAM inline policy to the role. For more information, see (Optional) add an Automation inline policy to invoke other AWS services (p. 284).

Topics
• Properties shared by all actions (p. 387)
• aws:approve – Pause an execution for manual approval (p. 392)
• aws:assertAwsResourceProperty – Assert an AWS resource state or event state (p. 395)
• aws:branch – Run conditional automation steps (p. 397)
• aws:changeInstanceState – Change or assert instance state (p. 399)
• aws:copyImage – Copy or encrypt an Amazon Machine Image (p. 401)
• aws:createImage – Create an Amazon Machine Image (p. 402)
• aws:createStack – Create an AWS CloudFormation stack (p. 404)
• aws:createTags – Create tags for AWS resources (p. 409)
• aws:deleteImage – Delete an Amazon Machine Image (p. 411)
• aws:deleteStack – Delete an AWS CloudFormation stack (p. 412)
• aws:executeAutomation – Run another automation execution (p. 413)
• aws:executeAwsApi – Call and run AWS API actions (p. 415)
• aws:executeScript – Run a script (p. 417)
• aws:executeStateMachine – Run an AWS Step Functions state machine (p. 420)
• aws:invokeLambdaFunction – Invoke an AWS Lambda function (p. 421)
• aws:pause – Pause an automation execution (p. 422)
• aws:runCommand – Run a command on a managed instance (p. 423)
• aws:runInstances – Launch an EC2 instance (p. 427)
• aws:sleep – Delay an automation execution (p. 431)
• aws:waitForAwsResourceProperty – Wait on an AWS resource property (p. 432)

Properties shared by all actions
Common properties are parameters or options that are found in all actions. Some options define execution behavior for a step, such as how long to wait for a step to complete and what to do if the step fails. The following properties are common to all actions.

name (p. 389)
An identifier that must be unique across all step names in the document.

Type: String
Required: Yes
action (p. 389)

The name of the action the step is to run. `aws:runCommand` – Run a command on a managed instance (p. 423) is an example of an action you can specify here. This document provides detailed information about all available actions.

Type: String
Required: Yes

maxAttempts (p. 389)

The number of times the step should be retried in case of failure. If the value is greater than 1, the step is not considered to have failed until all retry attempts have failed. The default value is 1.

Type: Integer
Required: No

timeoutSeconds (p. 390)

The execution timeout value for the step. If the timeout is reached and the value of `maxAttempts` is greater than 1, then the step is not considered to have timed out until all retries have been attempted.

The `aws:changeInstanceState` action has a default `timeoutSeconds` value of 3600. For all other actions, there is no default value.

Type: Integer
Required: No

onFailure (p. 390)

Indicates whether the workflow should abort, continue, or go to a different step on failure. The default value for this option is abort.

Type: String
Valid values: Abort | Continue | step:step_name
Required: No

isEnd (p. 390)

This option stops an Automation execution at the end of a specific step. The Automation execution stops if the step execution failed or succeeded. The default value is false.

Type: Boolean
Valid values: true | false
Required: No

nextStep (p. 390)

Specifies which step in an Automation workflow to process next after successfully completing a step.

Type: String
Required: No

isCritical (p. 391)

Designates a step as critical for the successful completion of the Automation. If a step with this designation fails, then Automation reports the final status of the Automation as Failed. The default value for this option is true.

Type: Boolean
Valid values: true | false

Required: No

inputs (p. 392)

The properties specific to the action.

Type: Map

Required: Yes

Example

```yaml
---
description: "Custom Automation Example"
schemaVersion: '0.3'
assumeRole: "{{ AutomationAssumeRole }}"
parameters:
  AutomationAssumeRole:
    type: String
    description: "(Required) The ARN of the role that allows Automation to perform
the actions on your behalf. If no role is specified, Systems Manager Automation
uses your IAM permissions to execute this document."
    default: ''
  InstanceId:
    type: String
    description: "(Required) The Instance Id whose root EBS volume you want to restore
the latest Snapshot."
    default: ''
mainSteps:
- name: getInstanceDetails
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeInstances
    InstanceIds:
      - "{{ InstanceId }}"
  outputs:
    - Name: availabilityZone
      Selector: "$.Reservations[0].Instances[0].Placement.AvailabilityZone"
      Type: String
    - Name: rootDeviceName
      Selector: "$.Reservations[0].Instances[0].RootDeviceName"
      Type: String
  nextStep: getRootVolumeId
- name: getRootVolumeId
  action: aws:executeAwsApi
  maxAttempts: 3
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeVolumes
    Filters:
      - Name: attachment.device
        Values: ["{{ getInstanceDetails.rootDeviceName }}"]
      - Name: attachment.instance-id
        Values: ["{{ InstanceId }}"]
  outputs:
    - Name: rootVolumeId
      Selector: "$.Volumes[0].VolumeId"
      Type: String
  nextStep: getSnapshotsByStartTime
- name: getSnapshotsByStartTime
```
action: aws:executeScript
timeoutSeconds: 45
onFailure: Abort
inputs:
  Runtime: python3.6
  Handler: getSnapshotsByStartTime
  InputPayload:
    rootVolumeId : "{{ getRootVolumeId.rootVolumeId }}"
  Script: |
    def getSnapshotsByStartTime(events,context):
      import boto3
      #Initialize client
      ec2 = boto3.client('ec2')
      rootVolumeId = events['rootVolumeId']
      snapshotsQuery = ec2.describe_snapshots(
        Filters=[
          {
            "Name": "volume-id",
            "Values": [rootVolumeId]
          }
        ]
      )
      if not snapshotsQuery['Snapshots']:
        noSnapshotFoundString = "NoSnapshotFound"
        return { 'noSnapshotFound' : noSnapshotFoundString }
      else:
        jsonSnapshots = snapshotsQuery['Snapshots']
        sortedSnapshots = sorted(jsonSnapshots, key=lambda k: k['StartTime'], reverse=True)
        latestSortedSnapshotId = sortedSnapshots[0]['SnapshotId']
        return { 'latestSnapshotId' : latestSortedSnapshotId }
outputs:
  - Name: Payload
    Selector: $.Payload
    Type: StringMap
  - Name: latestSnapshotId
    Selector: $.Payload.latestSnapshotId
    Type: String
  - Name: noSnapshotFound
    Selector: $.Payload.noSnapshotFound
    Type: String
nextStep: branchFromResults
- name: branchFromResults
  action: aws:branch
  onFailure: Abort
  inputs:
    Choices:
    - NextStep: createNewRootVolumeFromSnapshot
      Not:
        Variable: "{{getSnapshotsByStartTime.noSnapshotFound}}"
        StringEquals: "NoSnapshotFound"
        isEnd: true
    - name: createNewRootVolumeFromSnapshot
      action: aws:executeAwsApi
      onFailure: Abort
      inputs:
        Service: ec2
        Api: CreateVolume
        AvailabilityZone: "{{getInstanceDetails.availabilityZone}}"
        SnapshotId: "{{getSnapshotsByStartTime.latestSnapshotId}}"
        outputs:
          - Name: newRootVolumeId
            Selector: "$.VolumeId"
            Type: String
nextStep: stopInstance
- name: stopInstance
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: StopInstances
    InstanceIds:
      - "{{ InstanceId }}"
  nextStep: verifyVolumeAvailability
- name: verifyVolumeAvailability
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 120
  inputs:
    Service: ec2
    Api: DescribeVolumes
    VolumeIds:
      - "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
    PropertySelector: ".Volumes[0].State"
    DesiredValues:
      - "available"
  nextStep: verifyInstanceStopped
- name: verifyInstanceStopped
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 120
  inputs:
    Service: ec2
    Api: DescribeInstances
    InstanceIds:
      - "{{ InstanceId }}"
    PropertySelector: ".Reservations[0].Instances[0].State.Name"
    DesiredValues:
      - "stopped"
  nextStep: detachRootVolume
- name: detachRootVolume
  action: aws:executeAwsApi
  onFailure: Abort
  isCritical: true
  inputs:
    Service: ec2
    Api: DetachVolume
    VolumeId: "{{ getRootVolumeId.rootVolumeId }}"
  nextStep: verifyRootVolumeDetached
- name: verifyRootVolumeDetached
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 30
  inputs:
    Service: ec2
    Api: DescribeVolumes
    VolumeIds:
      - "{{ getRootVolumeId.rootVolumeId }}"
    PropertySelector: "$.Volumes[0].State"
    DesiredValues:
      - "available"
  nextStep: attachNewRootVolume
- name: attachNewRootVolume
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: AttachVolume
    Device: "{{ getInstanceDetails.rootDeviceName }}"
    InstanceId: "{{ InstanceId }}"
    VolumeId: "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
  nextStep: verifyNewRootVolumeAttached
- name: verifyNewRootVolumeAttached
  action: aws:waitForAwsResourceProperty
aws:approve – Pause an execution for manual approval

Temporarily pauses an Automation execution until designated principals either approve or reject the action. After the required number of approvals is reached, the Automation execution resumes. You can insert the approval step any place in the mainSteps section of your Automation document.

**Note**

The default timeout for this action is 7 days (604800 seconds). You can limit or extend the timeout by specifying the `timeoutSeconds` parameter for an `aws:approve` step. If the automation step reaches the timeout value before receiving all required approval decisions, then the step and the automation stop running and return a status of Timed Out.

In the following example, the `aws:approve` action temporarily pauses the Automation workflow until one approver either accepts or rejects the workflow. Upon approval, the document runs a simple PowerShell command.

YAML

```yaml
---
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:
    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

JSON

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

You can approve or deny Automations that are waiting for approval in the console.

**To approve or deny waiting Automations**

2. In the navigation pane, choose **Automation**.
   
   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon () to open the navigation pane, and then choose **Automation**.
3. Choose the option next to an Automation with a status of **Waiting**.

4. Choose **Approve/Deny**.

5. Review the details of the Automation.

6. Choose either **Approve** or **Deny**, type an optional comment, and then choose **Submit**.

**Input**

**YAML**

```yaml
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
```

**JSON**

```json
{
    "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**

The ARN of an Amazon SNS topic for Automation approvals. When you specify an `aws:approve` step in an Automation document, Automation sends a message to this topic letting principals know that they must either approve or reject an Automation step. The title of the Amazon SNS topic must be prefixed with “Automation”.

- **Type**: String
- **Required**: No

**Message**

The information you want to include in the SNS topic when the approval request is sent. The maximum message length is 4096 characters.

- **Type**: String
Required: No

**MinRequiredApprovals**

The minimum number of approvals required to resume the Automation execution. If you don't specify a value, the system defaults to one. The value for this parameter must be a positive number. The value for this parameter can't exceed the number of approvers defined by the `Approvers` parameter.

Type: Integer

Required: No

**Approvers**

A list of AWS authenticated principals who are able to either approve or reject the action. The maximum number of approvers is 10. You can specify principals by using any of the following formats:

- An AWS Identity and Access Management (IAM) user name
- An IAM user ARN
- An IAM role ARN
- An IAM assume role user ARN

Type: StringList

Required: Yes

**Output**

**ApprovalStatus**

The approval status of the step. The status can be one of the following: Approved, Rejected, or Waiting. Waiting means that Automation is waiting for input from approvers.

Type: String

**ApproverDecisions**

A JSON map that includes the approval decision of each approver.

Type: MapList

.aws:assertAwsResourceProperty – Assert an AWS resource state or event state

The `.aws:assertAwsResourceProperty` action enables you to assert a specific resource state or event state for a specific Automation step. For example, you can specify that an Automation step must wait for an EC2 instance to start. Then it will call the Amazon EC2 `DescribeInstanceStatus` API action with the `DesiredValue` property of `running`. This ensures that the Automation workflow waits for a running instance and then continues when the instance is, in fact, running.

For more information and examples of how to use this action, see Invoking other AWS services from a Systems Manager Automation workflow (p. 376).

**Note**

The default timeout value for this action is 3600 seconds (one hour). You can limit or extend the timeout by specifying the `timeoutSeconds` parameter for an `.aws:waitForAwsResourceProperty` step.

**Input**

Inputs are defined by the API action that you choose.
**YAML**

```
action: aws:assertAwsResourceProperty
inputs:
  Service: The official namespace of the service
  Api: The API action or method name
  API action inputs or parameters: A value
  PropertySelector: Response object
  DesiredValues:
    - Desired property values
```

**JSON**

```
{
  "action": "aws:assertAwsResourceProperty",
  "inputs": {
    "Service": "The official namespace of the service",
    "Api": "The API action or method name",
    "API action inputs or parameters": "A value",
    "PropertySelector": "Response object",
    "DesiredValues": [
      "Desired property values"
    ]
  }
}
```

**Service**

The AWS service namespace that contains the API action that you want to run. For example, the namespace for Systems Manager is `ssm`. The namespace for Amazon EC2 is `ec2`. You can view a list of supported AWS service namespaces in the Available Services section of the AWS CLI Command Reference.

Type: String

Required: Yes

**Api**

The name of the API action that you want to run. You can view the API actions (also called methods) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all API actions (methods) for Amazon Relational Database Service (Amazon RDS) are listed on the following page: Amazon RDS methods.

Type: String

Required: Yes

**API action inputs**

One or more API action inputs. You can view the available inputs (also called parameters) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all methods for Amazon RDS are listed on the following page: Amazon RDS methods. Choose the describe_db_instances method and scroll down to see the available parameters, such as `DBInstanceIdentifier`, `Name`, and `Values`. Use the following format to specify more than one input.

**YAML**

```
inputs:
```

**JSON**

```
{
  "inputs": {
    ...
  }
}
```
Service: The official namespace of the service
Api: The API action name
API input 1: A value
API Input 2: A value
API Input 3: A value

JSON

```
"inputs":{
    "Service":"The official namespace of the service",
    "Api":"The API action name",
    "API input 1":"A value",
    "API Input 2":"A value",
    "API Input 3":"A value"
}
```

Type: Determined by chosen API action
Required: Yes

PropertySelector

The JSONPath to a specific attribute in the response object. You can view the response objects by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all methods for Amazon RDS are listed on the following page: Amazon RDS methods. Choose the describe_db_instances method and scroll down to the Response Structure section. DBInstances is listed as a response object.

Type: Integer, Boolean, String, StringList, StringMap, or MapList
Required: Yes

DesiredValues

The expected status or state on which to continue the Automation workflow. If you specify a Boolean value, you must use a capital letter such as True or False.

Type: Varies
Required: Yes

aws:branch – Run conditional automation steps

The aws:branch action enables you to create a dynamic Automation workflow that evaluates different choices in a single step and then jumps to a different step in the Automation document based on the results of that evaluation.

When you specify the aws:branch action for a step, you specify Choices that the workflow must evaluate. The Choices can be based on either a value that you specified in the Parameters section of the Automation document, or a dynamic value generated as the output from the previous step. The Automation workflow evaluates each choice by using a Boolean expression. If the first choice is true, then the workflow jumps to the step designated for that choice. If the first choice is false, the workflow evaluates the next choice. The workflow continues evaluating each choice until it process a true choice. The workflow then jumps to the designated step for the true choice.

If none of the choices are true, the workflow checks to see if the step contains a default value. A default value defines a step that the workflow should jump to if none of the choices are true. If no default value is specified for the step, then the Automation workflow processes the next step in the document.
The `aws:branch` action supports complex choice evaluations by using a combination of `And`, `Not`, and `Or` operators. For more information about how to use `aws:branch`, including example documents and examples that use different operators, see Creating dynamic Automation workflows with conditional branching (p. 365).

**Input**

Specify one or more **Choices** in a step. The Choices can be based on either a value that you specified in the **Parameters** section of the Automation document, or a dynamic value generated as the output from the previous step. Here is a YAML sample that evaluates a parameter.

```yaml
mainSteps:
  - name: chooseOS
    action: aws:branch
    inputs:
      Choices:
        - NextStep: runWindowsCommand
          Variable: "{{Name of a parameter defined in the Parameters section. For example: OS_name}}"
          StringEquals: windows
        - NextStep: runLinuxCommand
          Variable: "{{Name of a parameter defined in the Parameters section. For example: OS_name}}"
          StringEquals: linux
        Default:
          sleep3
```

Here is a YAML sample that evaluates output from a previous step.

```yaml
mainSteps:
  - name: chooseOS
    action: aws:branch
    inputs:
      Choices:
        - NextStep: runPowerShellCommand
          Variable: "{{Name of a response object. For example: GetInstance.platform}}"
          StringEquals: Windows
        - NextStep: runShellCommand
          Variable: "{{Name of a response object. For example: GetInstance.platform}}"
          StringEquals: Linux
        Default:
          sleep3
```

**Choices**

One or more expressions that the Automation should evaluate when determining the next step to process. Choices are evaluated by using a Boolean expression. Each choice must define the following options:

- **NextStep**: The next step in the Automation document to process if the designated choice is true.
- **Variable**: Specify either the name of a parameter that is defined in the **Parameters** section of the Automation document. Or specify an output object from a previous step in the Automation document. For more information about creating variables for `aws:branch`, see About creating the output variable (p. 368).
- **Operation**: The criteria used to evaluate the choice. The `aws:branch` action supports the following operations:

  **String operations**
  - StringEquals
  - EqualsIgnoreCase
• StartsWith
• EndsWith
• Contains

**Numeric operations**
• NumericEquals
• NumericGreater
• NumericLesser
• NumericGreaterOrEquals
• NumericLesser
• NumericLesserOrEquals

**Boolean operation**
• BooleanEquals

**Important**
When you create an Automation document, the system validates each operation in the document. If an operation is not supported, the system returns an error when you try to create the document.

**Default**
The name of a step the workflow should jump to if none of the Choices are true.
Type: String
Required: No

**Note**
The `aws:branch` action supports And, Or, and Not operators. For examples of `aws:branch` that use operators, see Creating dynamic Automation workflows with conditional branching (p. 365).

**aws:changeInstanceState – Change or assert instance state**
Changes or asserts the state of the instance.

This action can be used in assert mode (do not run the API to change the state but verify the instance is in the desired state.) To use assert mode, set the `CheckStateOnly` parameter to true. This mode is useful when running the Sysprep command on Windows, which is an asynchronous command that can run in the background for a long time. You can ensure that the instance is stopped before you create an AMI.

**Note**
The default timeout value for this action is 3600 seconds (one hour). You can limit or extend the timeout by specifying the `timeoutSeconds` parameter for an `aws:changeInstanceState` step.

**Input**

YAML

```yaml
name: stopMyInstance
action: aws:changeInstanceState
maxAttempts: 3
timeoutSeconds: 3600
onFailure: Abort
```
inputs:
    InstanceIds:
    - i-1234567890abcdef0
    CheckStateOnly: true
    DesiredState: stopped

JSON

```
{
  "name": "stopMyInstance",
  "action": "aws:changeInstanceState",
  "maxAttempts": 3,
  "timeoutSeconds": 3600,
  "onFailure": "Abort",
  "inputs": {
    "InstanceIds": ["i-1234567890abcdef0"],
    "CheckStateOnly": true,
    "DesiredState": "stopped"
  }
}
```

InstanceIds

The IDs of the instances.

Type: StringList

Required: Yes

CheckStateOnly

If false, sets the instance state to the desired state. If true, asserts the desired state using polling.

Default: false

Type: Boolean

Required: No

DesiredState

The desired state. When set to running, this action waits for the Amazon EC2 state to be Running, the Instance Status to be OK, and the System Status to be OK before completing.

Type: String

Valid values: running | stopped | terminated

Required: Yes

Force

If set, forces the instances to stop. The instances do not have an opportunity to flush file system caches or file system metadata. If you use this option, you must perform file system check and repair procedures. This option is not recommended for EC2 instances for Windows Server.

Type: Boolean

Required: No

AdditionalInfo

Reserved.
Type: String
Required: No

Output
None

aws:copyImage – Copy or encrypt an Amazon Machine Image

Copies an AMI from any region into the current region. This action can also encrypt the new AMI.

Input
This action supports most CopyImage parameters. For more information, see CopyImage.

The following example creates a copy of an AMI in the Seoul region (SourceImageID: ami-0fe10819, SourceRegion: ap-northeast-2). The new AMI is copied to the region where you initiated the Automation action. The copied AMI will be encrypted because the optional Encrypted flag is set to true.

YAML

```yaml
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
```

JSON

```json
{
  "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

The region where the source AMI currently exists.

Type: String
Required: Yes

SourceImageId

The AMI ID to copy from the source region.

Type: String
Required: Yes

ImageName
The name for the new image.
Type: String
Required: Yes

ImageDescription
A description for the target image.
Type: String
Required: No

Encrypted
Encrypt the target AMI.
Type: Boolean
Required: No

KmsKeyId
The full Amazon Resource Name (ARN) of the AWS Key Management Service CMK to use when encrypting the snapshots of an image during a copy operation. For more information, see CopyImage.
Type: String
Required: No

ClientToken
A unique, case-sensitive identifier that you provide to ensure request idempotency. For more information, see CopyImage.
Type: String
Required: No

Output

ImageId
The ID of the copied image.

ImageState
The state of the copied image.
Valid values: available | pending | failed

aws:createImage – Create an Amazon Machine Image

Creates a new AMI from an instance that is either running or stopped.

Input

This action supports most CreateImage parameters. For more information, see CreateImage.
YAML

```
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
```

JSON

```
{
  "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"
  }
}
```

**InstanceId**

The ID of the instance.

Type: String

Required: Yes

**ImageName**

The name for the image.

Type: String

Required: Yes

**ImageDescription**

A description of the image.

Type: String

Required: No

**NoReboot**

A boolean literal.

By default, Amazon EC2 attempts to shut down and reboot the instance before creating the image. If the No Reboot option is set to true, Amazon EC2 doesn't shut down the instance before creating the image. When this option is used, file system integrity on the created image can't be guaranteed.

If you do not want the instance to run after you create an AMI image from it, first use the `aws:changeInstanceState` action to stop the instance, and then use this `aws:createImage` action with the NoReboot option set to true.
Type: Boolean
Required: No

BlockDeviceMappings
The block devices for the instance.
Type: Map
Required: No

Output
ImageId
The ID of the newly created image.

ImageState
The current state of the image. If the state is available, the image is successfully registered and can be used to launch an instance.

aws:createStack – Create an AWS CloudFormation stack

Creates a new AWS CloudFormation stack from a template.

For supplemental information about creating AWS CloudFormation stacks, see CreateStack in the AWS CloudFormation API Reference.

Input

YAML

name: makeStack
action: aws:createStack
maxAttempts: 1
onFailure: Abort
inputs:
 Capabilities:
  - CAPABILITY_IAM
StackName: myStack
TemplateURL: http://s3.amazonaws.com/mybucket/myStackTemplate
TimeoutInMinutes: 5

JSON

{  "name": "makeStack",
  "action": "aws:createStack",
  "maxAttempts": 1,
  "onFailure": "Abort",
  "inputs": {
    "Capabilities": [
      "CAPABILITY_IAM"
    ],
    "StackName": "myStack",
    "TemplateURL": "http://s3.amazonaws.com/mybucket/myStackTemplate",
    "TimeoutInMinutes": 5
  }
}
Capabilities

A list of values that you specify before AWS CloudFormation can create certain stacks. Some stack templates include resources that can affect permissions in your AWS account. For example, creating new AWS Identity and Access Management (IAM) users can affect permissions in your account. For those stacks, you must explicitly acknowledge their capabilities by specifying this parameter.

Valid values include CAPABILITY_IAM, CAPABILITY_NAMED_IAM, and CAPABILITY_AUTO_EXPAND.

**CAPABILITY_IAM and CAPABILITY_NAMED_IAM**

If you have IAM resources, you can specify either capability. If you have IAM resources with custom names, you must specify CAPABILITY_NAMED_IAM. If you don't specify this parameter, this action returns an InsufficientCapabilities error. The following resources require you to specify either CAPABILITY_IAM or CAPABILITY_NAMED_IAM.

- AWS::IAM::AccessKey
- AWS::IAM::Group
- AWS::IAM::InstanceProfile
- AWS::IAM::Policy
- AWS::IAM::Role
- AWS::IAM::User
- AWS::IAM::UserToGroupAddition

If your stack template contains these resources, we recommend that you review all permissions associated with them and edit their permissions, if necessary.

For more information, see Acknowledging IAM Resources in AWS CloudFormation Templates.

**CAPABILITY_AUTO_EXPAND**

Some template contain macros. Macros perform custom processing on templates; this can include simple actions like find-and-replace operations, all the way to extensive transformations of entire templates. Because of this, users typically create a change set from the processed template, so that they can review the changes resulting from the macros before actually creating the stack. If your stack template contains one or more macros, and you choose to create a stack directly from the processed template, without first reviewing the resulting changes in a change set, you must acknowledge this capability.

For more information, see Using AWS CloudFormation Macros to Perform Custom Processing on Templates in the AWS CloudFormation User Guide.

Type: array of Strings

Valid Values: CAPABILITY_IAM | CAPABILITY_NAMED_IAM | CAPABILITY_AUTO_EXPAND

Required: No

ClientRequestToken

A unique identifier for this CreateStack request. Specify this token if you set maxAttempts in this step to a value greater than 1. By specifying this token, AWS CloudFormation knows that you're not attempting to create a new stack with the same name.

Type: String

Required: No

Pattern: [a-zA-Z0-9][-a-zA-Z0-9]*

DisableRollback

Set to true to disable rollback of the stack if stack creation failed.

Conditional: You can specify either the DisableRollback parameter or the OnFailure parameter, but not both.

Default: false

Type: Boolean

Required: No

NotificationARNs

The Amazon SNS topic ARNs for publishing stack-related events. You can find SNS topic ARNs using the Amazon SNS console, https://console.aws.amazon.com/sns/v3/home.

Type: array of Strings

Array Members: Maximum number of 5 items.

Required: No

OnFailure

Determines the action to take if stack creation failed. You must specify DO NOTHING, ROLLBACK, or DELETE.

Conditional: You can specify either the OnFailure parameter or the DisableRollback parameter, but not both.

Default: ROLLBACK

Type: String

Valid Values: DO NOTHING | ROLLBACK | DELETE

Required: No

Parameters

A list of Parameter structures that specify input parameters for the stack. For more information, see the Parameter data type.

Type: array of Parameter objects

Required: No

ResourceTypes

The template resource types that you have permissions to work with for this create stack action. For example: AWS::EC2::Instance, AWS::EC2::* , or Custom::MyCustomInstance. Use the following syntax to describe template resource types.

- For all AWS resources:

```plaintext
AWS::*
```

- For all custom resources:

```plaintext
Custom::*
```

- For a specific custom resource:
Custom::**logical_ID**

- For all resources of a particular AWS service:

  **AWS::service_name::*

- For a specific AWS resource:

  **AWS::service_name::resource_logical_ID**

If the list of resource types doesn't include a resource that you're creating, the stack creation fails. By default, AWS CloudFormation grants permissions to all resource types. IAM uses this parameter for AWS CloudFormation-specific condition keys in IAM policies. For more information, see **Controlling Access with AWS Identity and Access Management**.

Type: array of Strings

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

RoleARN

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes to create the stack. AWS CloudFormation uses the role's credentials to make calls on your behalf. AWS CloudFormation always uses this role for all future operations on the stack. As long as users have permission to operate on the stack, AWS CloudFormation uses this role even if the users don't have permission to pass it. Ensure that the role grants the least amount of privileges.

If you don't specify a value, AWS CloudFormation uses the role that was previously associated with the stack. If no role is available, AWS CloudFormation uses a temporary session that is generated from your user credentials.

Type: String


Required: No

StackName

The name that is associated with the stack. The name must be unique in the region in which you are creating the stack.

**Note**

A stack name can contain only alphanumeric characters (case sensitive) and hyphens. It must start with an alphabetic character and cannot be longer than 128 characters.

Type: String

Required: Yes

StackPolicyBody

Structure containing the stack policy body. For more information, see **Prevent Updates to Stack Resources**.

Conditional: You can specify either the StackPolicyBody parameter or the StackPolicyURL parameter, but not both.

Type: String

Required: No

StackPolicyURL

Location of a file containing the stack policy. The URL must point to a policy located in an S3 bucket in the same region as the stack. The maximum file size allowed for the stack policy is 16 KB.

Conditional: You can specify either the StackPolicyBody parameter or the StackPolicyURL parameter, but not both.

Type: String


Required: No

Tags

Key-value pairs to associate with this stack. AWS CloudFormation also propagates these tags to the resources created in the stack. You can specify a maximum number of 10 tags.

Type: array of Tag objects

Required: No

TemplateBody

Structure containing the template body with a minimum length of 1 byte and a maximum length of 51,200 bytes. For more information, see Template Anatomy.

Conditional: You can specify either the TemplateBody parameter or the TemplateURL parameter, but not both.

Type: String

Length Constraints: Minimum length of 1.

Required: No

TemplateURL

Location of a file containing the template body. The URL must point to a template that is located in an S3 bucket. The maximum size allowed for the template is 460,800 bytes. For more information, see Template Anatomy.

Conditional: You can specify either the TemplateBody parameter or the TemplateURL parameter, but not both.

Type: String


Required: No

TimeoutInMinutes

The amount of time that can pass before the stack status becomes CREATE_FAILED. If DisableRollback is not set or is set to false, the stack will be rolled back.

Type: Integer

Valid Range: Minimum value of 1.

Required: No
Outs

StackId

Unique identifier of the stack.

Type: String

StackStatus

Current status of the stack.

Type: String

Valid Values:

- 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

Required: Yes

StackStatusReason

Success or failure message associated with the stack status.

Type: String

Required: No

For more information, see CreateStack.

Security considerations

Before you can use the aws:createStack action, you must assign the following policy to the IAM Automation assume role. For more information about the assume role, see Task 1: Create a service role for Automation (p. 283).

```json
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect":"Allow",
      "Action": [
        "sqs:*",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStacks"
      ],
      "Resource":"**"
    }
  ]
}
```

aws:createTags – Create tags for AWS resources

Create new tags for EC2 instances or Systems Manager managed instances.

Input

This action supports most EC2 CreateTags and SSM AddTagsToResource parameters. For more information, see CreateTags and AddTagsToResource.
The following example shows how to tag an AMI and an instance as being production resources for a particular department.

**YAML**

```yaml
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
```

**JSON**

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

**ResoureIds**

The IDs of the resource(s) to be tagged. If resource type is not "EC2", this field can contain only a single item.

Type: String List

Required: Yes

**Tags**

The tags to associate with the resource(s).

Type: List of Maps

Required: Yes
ResourceType

The type of resource(s) to be tagged. If not supplied, the default value of “EC2” is used.

Type: String
Required: No

Valid Values: EC2 | ManagedInstance | MaintenanceWindow | Parameter

Output

None

aws:deleteImage – Delete an Amazon Machine Image

Deletes the specified image and all related snapshots.

Input

This action supports only one parameter. For more information, see the documentation for DeregisterImage and DeleteSnapshot.

YAML

```yaml
name: deleteMyImage
action: aws:deleteImage
maxAttempts: 3
timeoutSeconds: 180
onFailure: Abort
inputs:
  ImageId: ami-12345678
```

JSON

```json
{
  "name": "deleteMyImage",
  "action": "aws:deleteImage",
  "maxAttempts": 3,
  "timeoutSeconds": 180,
  "onFailure": "Abort",
  "inputs": {
    "ImageId": "ami-12345678"
  }
}
```

Input

The ID of the image to be deleted.

Type: String
Required: Yes

Output

None
**aws:deleteStack – Delete an AWS CloudFormation stack**

Deletes an AWS CloudFormation stack.

**Input**

**YAML**

```yaml
name: deleteStack
action: aws:deleteStack
maxAttempts: 1
onFailure: Abort
inputs:
  StackName: "{{stackName}}"
```

**JSON**

```json
{
  "name": "deleteStack",
  "action": "aws:deleteStack",
  "maxAttempts": 1,
  "onFailure": "Abort",
  "inputs": {
    "StackName": "{{stackName}}"
  }
}
```

**ClientRequestToken**

A unique identifier for this DeleteStack request. Specify this token if you plan to retry requests so that AWS CloudFormation knows that you're not attempting to delete a stack with the same name. You can retry DeleteStack requests to verify that AWS CloudFormation received them.

Type: String


Pattern: [a-zA-Z][a-zA-Z0-9]*

Required: No

**RetainResources.member.N**

This input applies only to stacks that are in a DELETE_FAILED state. A list of logical resource IDs for the resources you want to retain. During deletion, AWS CloudFormation deletes the stack, but does not delete the retained resources.

Retaining resources is useful when you can't delete a resource, such as a non-empty S3 bucket, but you want to delete the stack.

Type: array of strings

Required: No

**RoleARN**

The Amazon Resource Name (ARN) of an IAM role that AWS CloudFormation assumes to create the stack. AWS CloudFormation uses the role's credentials to make calls on your behalf. AWS CloudFormation always uses this role for all future operations on the stack. As long as users have permission to operate on the stack, AWS CloudFormation uses this role even if the users don't have permission to pass it. Ensure that the role grants the least amount of privileges.
If you don't specify a value, AWS CloudFormation uses the role that was previously associated with the stack. If no role is available, AWS CloudFormation uses a temporary session that is generated from your user credentials.

Type: String


Required: No

StackName

The name or the unique stack ID that is associated with the stack.

Type: String

Required: Yes

Security considerations

Before you can use the `aws:deleteStack` action, you must assign the following policy to the IAM Automation assume role. For more information about the assume role, see Task 1: Create a service role for Automation (p. 283).

```json
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect":"Allow",
         "Action": ["sqs:*", "cloudformation:DeleteStack", "cloudformation:DescribeStacks"],
         "Resource": "*"
      }
   ]
}
```

`aws:executeAutomation` – Run another automation execution

Runs a secondary Automation workflow by calling a secondary Automation document. With this action, you can create Automation documents for your most common workflows, and reference those documents during an Automation execution. This action can simplify your Automation documents by removing the need to duplicate steps across similar documents.

The secondary Automation runs in the context of the user who initiated the primary Automation. This means that the secondary Automation uses the same IAM role or user account as the user who started the first Automation.

**Important**

If you specify parameters in a secondary Automation that use an assume role (a role that uses the `iam:passRole` policy), then the user or role that initiated the primary Automation must have permission to pass the assume role specified in the secondary Automation. For more information about setting up an assume role for Automation, see Method 2: Use IAM to configure roles for Automation (p. 283).

Input

YAML

```
name: Secondary_Automation_Workflow
```
action: aws:executeAutomation
maxAttempts: 3
timeoutSeconds: 3600
onFailure: Abort
inputs:
  DocumentName: secondaryWorkflow
  RuntimeParameters:
    instanceIds:
      - i-1234567890abcdef0

JSON

```
{
  "name":"Secondary_Automation_Workflow",
  "action":"aws:executeAutomation",
  "maxAttempts":3,
  "timeoutSeconds":3600,
  "onFailure":"Abort",
  "inputs":{
    "DocumentName":"secondaryWorkflow",
    "RuntimeParameters":{
      "instanceIds":[
        "i-1234567890abcdef0"
      ]
    }
  }
}
```

DocumentName

The name of the secondary Automation document to run during the step. The document must belong to the same AWS account as the primary Automation document.

Type: String
Required: Yes

DocumentVersion

The version of the secondary Automation document to run. If not specified, Automation runs the default document version.

Type: String
Required: Yes

RuntimeParameters

Required parameters for the secondary document execution. The mapping uses the following format: 
{"parameter1": ["value1"], "parameter2": ["value2"]}

Type: Map
Required: No

Output

Output

The output generated by the secondary execution. You can reference the output by using the following format: Secondary_Automation_Step_Name.Output
Type: StringList

Here is an example:

```plaintext
- name: launchNewWindowsInstance
  action: 'aws:executeAutomation'
  onFailure: Abort
  inputs:
    DocumentName: launchWindowsInstance
  nextStep: getNewInstanceRootVolume
- name: getNewInstanceRootVolume
  action: 'aws:executeAwsApi'
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeVolumes
    Filters:
    - Name: attachment.device
      Values:
      - /dev/sda1
    - Name: attachment.instance-id
      Values:
      - '{{launchNewWindowsInstance.Output}}'
  outputs:
    - Name: rootVolumeId
      Selector: '$.Volumes[0].VolumeId'
      Type: String
  nextStep: snapshotRootVolume
- name: snapshotRootVolume
  action: 'aws:executeAutomation'
  onFailure: Abort
  inputs:
    DocumentName: AWS-CreateSnapshot
    RuntimeParameters:
    VolumeId:
    - '{{getNewInstanceRootVolume.rootVolumeId}}'
    Description:
    - 'Initial root snapshot for {{launchNewWindowsInstance.Output}}'
```

**ExecutionId**

The execution ID of the secondary execution.

Type: String

**Status**

The status of the secondary execution.

Type: String

---

**aws:executeAwsApi — Call and run AWS API actions**

Calls and runs AWS API actions. Most API actions are supported, although not all API actions have been tested. For example, the following API actions are supported: `CreateImage`, `Delete bucket`, `RebootDBInstance`, and `CreateGroups`, to name a few. Streaming API actions, such as the `GetObject` action, aren’t supported. For more information and examples of how to use this action, see Invoking other AWS services from a Systems Manager Automation workflow (p. 376).

**Input**

Inputs are defined by the API action that you choose.
YAML

```yaml
action: aws:executeAwsApi
inputs:
  Service: The official namespace of the service
  Api: The API action or method name
  API action inputs or parameters: A value
outputs: # These are user-specified outputs
  - Name: The name for a user-specified output key
    Selector: A response object specified by using jsonpath format
    Type: The data type
```

JSON

```json
{
  "action":"aws:executeAwsApi",
  "inputs":{
    "Service":"The official namespace of the service",
    "Api":"The API action or method name",
    "API action inputs or parameters":"A value"
  },
  "outputs":{
    "These are user-specified outputs":{
      "Name":"The name for a user-specified output key",
      "Selector":"A response object specified by using JSONPath format",
      "Type":"The data type"
    }
  }
}
```

Service

The AWS service namespace that contains the API action that you want to run. You can view a list of supported AWS service namespaces in the Available services of the AWS SDK for Python (Boto 3). The namespace can be found in the Client section. For example, the namespace for Systems Manager is ssm. The namespace for Amazon EC2 is ec2.

Type: String

Required: Yes

Api

The name of the API action that you want to run. You can view the API actions (also called methods) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all API actions (methods) for Amazon RDS are listed on the following page: Amazon RDS methods.

Type: String

Required: Yes

API action inputs

One or more API action inputs. You can view the available inputs (also called parameters) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all methods for Amazon RDS are listed on the following page: Amazon RDS methods. Choose the describe_db_instances method and scroll down to see the available parameters, such as DBInstanceIdentifier, Name, and Values.
YAML

inputs:
  Service: The official namespace of the service
  Api: The API action name
  API Input 1: A value
  API Input 2: A value
  API Input 3: A value

JSON

"inputs": {
  "Service": "The official namespace of the service",
  "Api": "The API action name",
  "API Input 1": "A value",
  "API Input 2": "A value",
  "API Input 3": "A value"
}

Type: Determined by chosen API action
Required: Yes

Name
A name for the output.
Type: String
Required: Yes

Selector

The JSONPath to a specific attribute in the response object. You can view the response objects by choosing a service in the left navigation on the following Services Reference page. Choose a method in the
Client section for the service that you want to invoke. For example, all methods for Amazon RDS are listed on the following page: Amazon RDS methods. Choose the describe_db_instances method and scroll down to the Response Structure section. DBInstances is listed as a response object.
Type: Integer, Boolean, String, StringList, StringMap, or MapList
Required: Yes

Type
The data type for the response element.
Type: Varies
Required: Yes

aws:executeScript – Run a script

Runs the Python or PowerShell script provided, using the specified runtime and handler. (For PowerShell, the handler is not required.)

Currently, the aws:executeScript action contains the following preinstalled PowerShell Core modules.
• Microsoft.PowerShell.Host
To use PowerShell Core modules that are not preinstalled, your script must install the module with the `-Force` flag, as shown in the following command.

```
Install-Module ModuleName -Force
```

To use PowerShell Core cmdlets in your script, we recommend using the `AWS.Tools` modules, as shown in the following commands.

**Important**
Installing the `AWSPowerShell.NetCore` module is not supported.

- Amazon S3 cmdlets.

```
Install-Module AWS.Tools.S3 -Force
Get-S3Bucket -BucketName bucketname
```

- Amazon EC2 cmdlets.

```
Install-Module AWS.Tools.EC2 -Force
Get-EC2InstanceStatus -InstanceId instanceId
```

- Common, or service independent AWS Tools for Windows PowerShell cmdlets.

```
Install-Module AWS.Tools.Common -Force
Get-AWSRegion
```

If your script initializes new objects in addition to using PowerShell Core cmdlets, you must also import the module as shown in the following command.

```
Install-Module AWS.Tools.EC2 -Force
Import-Module AWS.Tools.EC2
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"
New-EC2Tag -Resource i-12345678 -Tag $tag
```

For examples of installing and importing `AWS.Tools` modules, and using PowerShell Core cmdlets in Automation document content, see `Walkthrough: Using Document Builder to create a custom Automation document (p. 604)`.

**Input**

Provide the runtime and handler required to run the provided Python 3.6, Python 3.7, or PowerShell Core 6.0 script.

**Important**
The script input parameter is not supported for JSON documents. JSON documents must provide script content using the attachment input parameter.
YAML

```
action: "aws:executeScript"
inputs:
  Runtime: "python3.6"
  Handler: "script_handler"
  InputPayload:
    "parameter1": "parameter_value1"
    "parameter2": "parameter_value2"
  Script:
    - "def script_handler(events, context):
    - "(script commands)"
Attachment: "zip-file-name-1.zip"
```

JSON

```json
{
  "action": "aws:executeScript",
  "inputs": {
    "Runtime": "python3.6",
    "Handler": "script_handler",
    "InputPayload": {
      "parameter1": "parameter_value1",
      "parameter2": "parameter_value2"
    },
    "Attachment": "zip-file-name-1.zip"
  }
}
```

Runtime

The runtime language to be used for executing the provided script. Currently, aws:executeScript supports Python 3.6 (python3.6), Python 3.7 (python3.7), and PowerShell Core 6.0 (dotnetcore2.1) scripts.

Supported values: python3.6 | python3.7 | PowerShell Core 6.0

Type: String

Required: Yes

Handler

The entry for script execution, usually a function name. You must ensure the function defined in the handler has two parameters, events and context. (Not required for PowerShell.)

Type: String

Required: Yes (Python) | No (PowerShell)

InputPayload

A JSON or YAML object that will be passed to the first parameter of the handler. This can be used to pass input data to the script.

Type: String

Required: No
Script

An embedded script that you want to run during the automation execution. (Not supported for JSON documents.)

Type: String

Required: No (Python) | Yes (PowerShell)

Attachment

The name of a standalone script file or .zip file that can be invoked by the action. To invoke a file for python, use the `filename.method_name` format in Handler. For PowerShell, invoke the attachment using and inline script. Gzip is not supported.

Type: String

Required: No

**aws:executeStateMachine – Run an AWS Step Functions state machine**

Run an AWS Step Functions state machine.

**Input**

This action supports most parameters for the Step Functions StartExecution API action.

**YAML**

```yaml
name: executeTheStateMachine
action: aws:executeStateMachine
inputs:
  stateMachineArn: StateMachine_ARN
  input: '{"parameters":"values"}
  name: name
```

**JSON**

```json
{
  "name": "executeTheStateMachine",
  "action": "aws:executeStateMachine",
  "inputs": {
    "stateMachineArn": "StateMachine_ARN",
    "input": `{"parameters":"values"}`,
    "name": "name"
  }
}
```

**stateMachineArn**

The ARN of the Step Functions state machine.

Type: String

Required: Yes

**name**

The name of the execution.

Type: String
Required: No

input

A string that contains the JSON input data for the execution.

Type: String

Required: No

aws:invokeLambdaFunction – Invoke an AWS Lambda function

Invokes the specified Lambda function.

Input

This action supports most invoked parameters for the Lambda service. For more information, see Invoke.

YAML

```yaml
name: invokeMyLambdaFunction
action: aws:invokeLambdaFunction
maxAttempts: 3
timeoutSeconds: 120
onFailure: Abort
inputs:
  FunctionName: MyLambdaFunction
```

JSON

```json
{
  "name": "invokeMyLambdaFunction",
  "action": "aws:invokeLambdaFunction",
  "maxAttempts": 3,
  "timeoutSeconds": 120,
  "onFailure": "Abort",
  "inputs": {
    "FunctionName": "MyLambdaFunction"
  }
}
```

FunctionName

The name of the Lambda function. This function must exist.

Type: String

Required: Yes

Qualifier

The function version or alias name.

Type: String

Required: No

InvocationType

The invocation type. The default is RequestResponse.

Type: String
Valid values: Event | RequestResponse | DryRun

Required: No

LogType

If Tail, the invocation type must be RequestResponse. AWS Lambda returns the last 4 KB of log data produced by your Lambda function, base64-encoded.

Type: String

Valid values: None | Tail

Required: No

ClientContext

The client-specific information.

Required: No

Payload

The JSON input for your Lambda function.

Required: No

Output

StatusCode

The function execution status code.

FunctionError

Indicates whether an error occurred while running the Lambda function. If an error occurred, this field will show either Handled or Unhandled. Handled errors are reported by the function. Unhandled errors are detected and reported by AWS Lambda.

LogResult

The base64-encoded logs for the Lambda function invocation. Logs are present only if the invocation type is RequestResponse, and the logs were requested.

Payload

The JSON representation of the object returned by the Lambda function. Payload is present only if the invocation type is RequestResponse.

aws:pause – Pause an automation execution

This action pauses the Automation execution. Once paused, the execution status is Waiting. To continue the Automation execution, use the SendAutomationSignal API action with the Resume signal type.

Input

The input is as follows.

YAML

```yaml
name: pauseThis
action: aws:pause
inputs: {}
```
### AWS Systems Manager User Guide

**Working with Automation documents (Playbooks)**

#### JSON

```json
{
    "name": "pauseThis",
    "action": "aws:pause",
    "inputs": {}
}
```

#### Output

None

### aws:runCommand – Run a command on a managed instance

Runs the specified commands.

**Note**

Automation only supports output of one Run Command action. A document can include multiple Run Command actions, but output is supported for only one action at a time.

#### Input

This action supports most send command parameters. For more information, see [SendCommand](#).

#### YAML

```yaml
- name: checkMembership
  action: 'aws:runCommand'
  inputs:
    DocumentName: AWS-RunPowerShellScript
    InstanceIds: 
      - '{{InstanceIds}}'
  Parameters:
    commands:
      - (Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain
```

#### JSON

```json
{
    "name": "checkMembership",
    "action": "aws:runCommand",
    "inputs": {
        "DocumentName": "AWS-RunPowerShellScript",
        "InstanceIds": [ 
            "{{InstanceIds}}"
        ],
        "Parameters": {
            "commands": [ 
                "(Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain"
            ]
        }
    }
}
```

#### DocumentName

The name of the Run Command document.
Type: String

Required: Yes

InstanceIds

The instance IDs where you want the command to run. You can specify a maximum of 50 IDs.

You can also use the pseudo parameter {{RESOURCE_ID}} in place of instance IDs to more easily run the command on all instances in the target group. For more information about pseudo parameters, see About pseudo parameters (p. 659).

Another alternative is to send commands to a fleet of instances by using the Targets parameter. The Targets parameter accepts Amazon EC2 tags. For more information about how to use the Targets parameter, see Using targets and rate controls to send commands to a fleet (p. 840).

Type: StringList

Required: No (If you don't specify InstanceIds or use the {{RESOURCE_ID}} pseudo parameter, then you must specify the Targets parameter.)

Targets

An array of search criteria that targets instances by using a Key,Value combination that you specify. Targets is required if you don't provide one or more instance IDs in the call. For more information about how to use the Targets parameter, see Using targets and rate controls to send commands to a fleet (p. 840).

Type: MapList (The schema of the map in the list must match the object. For information, see Target in the AWS Systems Manager API Reference.)

Required: No (If you don't specify Targets, then you must specify InstanceIds or use the {{RESOURCE_ID}} pseudo parameter.)

Here is an example:

YAML

```yaml
- name: checkMembership
  action: aws:runCommand
  inputs:
    DocumentName: AWS-RunPowerShellScript
    Targets:
    - Key: tag:Stage
      Values:
      - Gamma
      - Beta
    - Key: tag-key
      Values:
      - Suite
    Parameters:
    commands:
    - (Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain
```

JSON

```json
{
  "name": "checkMembership",
  "action": "aws:runCommand",
  "inputs": {
    "DocumentName": "AWS-RunPowerShellScript",
    "Targets": [
      {
        "Key": "tag:Stage",
```
Parameters

The required and optional parameters specified in the document.

Type: Map

Required: No

CloudWatchOutputConfig

Configuration options for sending command output to Amazon CloudWatch Logs. For more information about sending command output to CloudWatch Logs, see Configuring Amazon CloudWatch Logs for Run Command (p. 1146).

Type: StringMap (The schema of the map must match the object. For more information, see CloudWatchOutputConfig in the AWS Systems Manager API Reference).

Required: No

Here is an example:

YAML

```yaml
- name: checkMembership
  action: aws:runCommand
  inputs:
    DocumentName: AWS-RunPowerShellScript
    InstanceIds:
    - "{{InstanceIds}}"
  Parameters:
    commands:
    - "$(Get-WmiObject -Class Win32_ComputerSystem).PartOfDomain"
  CloudWatchOutputConfig:
    CloudWatchLogGroupName: CloudWatchGroupForSSMAutomationService
    CloudWatchOutputEnabled: true
```

JSON

```json
{
  "name": "checkMembership",
  "action": "aws:runCommand",
  "inputs": {
    "DocumentName": "AWS-RunPowerShellScript",
    "InstanceIds": ["{{InstanceIds}}"]
  }
}
```

Comment
User-defined information about the command.
Type: String
Required: No

DocumentHash
The hash for the document.
Type: String
Required: No

DocumentHashType
The type of the hash.
Type: String
Valid values: Sha256 | Sha1
Required: No

NotificationConfig
The configurations for sending notifications.
Required: No

OutputS3BucketName
The name of the S3 bucket for command execution responses.
Type: String
Required: No

OutputS3KeyPrefix
The prefix.
Type: String
Required: No

ServiceRoleArn
The ARN of the IAM role.
Type: String
Required: No
TimeoutSeconds
  The run-command timeout value, in seconds.
  Type: Integer
  Required: No

Output

CommandId
  The ID of the command.
Status
  The status of the command.
ResponseCode
  The response code of the command.
Output
  The output of the command.

aws:runInstances – Launch an EC2 instance
Launches a new instance.

Input
The action supports most API parameters. For more information, see the RunInstances API documentation.

YAML

```yaml
name: launchInstance
action: aws:runInstances
maxAttempts: 3
timeoutSeconds: 1200
onFailure: Abort
inputs:
  ImageId: ami-12345678
  InstanceType: t2.micro
  MinInstanceCount: 1
  MaxInstanceCount: 1
  IamInstanceProfileName: myRunCmdRole
TagSpecifications:
  - ResourceType: instance
    Tags:
      - Key: LaunchedBy
        Value: SSMAutomation
      - Key: Category
        Value: HighAvailabilityFleetHost
```

JSON

```json
{
  "name":"launchInstance",
  "action":"aws:runInstances",
  "maxAttempts":3,
  "maxAttempts":3,
  "timeoutSeconds":1200
}
"timeoutSeconds":1200,
"onFailure":"Abort",
"inputs":{
  "ImageId":"ami-12345678",
  "InstanceType":"t2.micro",
  "MinInstanceCount":1,
  "MaxInstanceCount":1,
  "IamInstanceProfileName":"myRunCmdRole",
  "TagSpecifications":[
    {
      "ResourceType":"instance",
      "Tags":[
        {
          "Key":"LaunchedBy",
          "Value":"SSMAutomation"
        },
        {
          "Key":"Category",
          "Value":"HighAvailabilityFleetHost"
        }
      ]
    }
  ]
}

ImageId

The ID of the Amazon Machine Image (AMI).

Type: String

Required: Yes

InstanceType

The instance type.

Note

If an instance type value is not provided, the m1.small instance type is used.

Type: String

Required: No

MinInstanceCount

The minimum number of instances to be launched.

Type: String

Required: No

MaxInstanceCount

The maximum number of instances to be launched.

Type: String

Required: No

AdditionalInfo

Reserved.

Type: String
Required: No
BlockDeviceMappings
   The block devices for the instance.
   Type: MapList
   Required: No
ClientToken
   The identifier to ensure idempotency of the request.
   Type: String
   Required: No
DisableApiTermination
   Enables or disables instance API termination
   Type: Boolean
   Required: No
EbsOptimized
   Enables or disabled EBS optimization.
   Type: Boolean
   Required: No
IamInstanceProfileArn
   The ARN of the IAM instance profile for the instance.
   Type: String
   Required: No
IamInstanceProfileName
   The name of the IAM instance profile for the instance.
   Type: String
   Required: No
InstanceInitiatedShutdownBehavior
   Indicates whether the instance stops or terminates on system shutdown.
   Type: String
   Required: No
KernelId
   The ID of the kernel.
   Type: String
   Required: No
KeyName
   The name of the key pair.
MaxInstanceCount
   The maximum number of instances to filter when searching for offerings.
   Type: Integer
   Required: No

MinInstanceCount
   The minimum number of instances to filter when searching for offerings.
   Type: Integer
   Required: No

Monitoring
   Enables or disables detailed monitoring.
   Type: Boolean
   Required: No

NetworkInterfaces
   The network interfaces.
   Type: MapList
   Required: No

Placement
   The placement for the instance.
   Type: StringMap
   Required: No

PrivateIpAddress
   The primary IPv4 address.
   Type: String
   Required: No

RamdiskId
   The ID of the RAM disk.
   Type: String
   Required: No

SecurityGroupIds
   The IDs of the security groups for the instance.
   Type: StringList
   Required: No

SecurityGroups
   The names of the security groups for the instance.
Type: StringList
Required: No

SubnetId

The subnet ID.
Type: String
Required: No

TagSpecifications

The tags to apply to the resources during launch. You can only tag instances and volumes at launch. The specified tags are applied to all instances or volumes that are created during launch. To tag an instance after it has been launched, use the `aws:createTags – Create tags for AWS resources (p. 409)` action.
Type: MapList (For more information, see TagSpecification.)
Required: No

UserData

An execution script provided as a string literal value. If a literal value is entered, then it must be Base64-encoded.
Type: String
Required: No

Output

InstanceIds

The IDs of the instances.

**aws:sleep – Delay an automation execution**

Delays Automation execution for a specified amount of time. This action uses the International Organization for Standardization (ISO) 8601 date and time format. For more information about this date and time format, see ISO 8601.

**Input**

You can delay execution for a specified duration.

**YAML**

```
name: sleep
action: aws:sleep
inputs:
  Duration: PT10M
```

**JSON**

```
{
  "name": "sleep",
  "action": "aws:sleep",
  "inputs": {
    "Duration": "PT10M"
  }
}```
You can also delay execution until a specified date and time. If the specified date and time has passed, the action proceeds immediately.

**YAML**

```yaml
name: sleep
action: aws:sleep
inputs:
  Timestamp: '2020-01-01T01:00:00Z'
```

**JSON**

```json
{
  "name": "sleep",
  "action": "aws:sleep",
  "inputs": {
    "Timestamp": "2020-01-01T01:00:00Z"
  }
}
```

**Note**

Automation currently supports a maximum delay of 604800 seconds (7 days).

**Duration**

An ISO 8601 duration. You can't specify a negative duration.

Type: String

Required: No

**Timestamp**

An ISO 8601 timestamp. If you don't specify a value for this parameter, then you must specify a value for the **Duration** parameter.

Type: String

Required: No

**Output**

None

**aws:waitForAwsResourceProperty – Wait on an AWS resource property**

The `aws:waitForAwsResourceProperty` action enables your Automation workflow to wait for a specific resource state or event state before continuing the workflow. For more information and examples of how to use this action, see Invoking other AWS services from a Systems Manager Automation workflow (p. 376).

**Input**

Inputs are defined by the API action that you choose.
YAML

```
action: aws:waitForAwsResourceProperty
inputs:
  Service: The official namespace of the service
  Api: The API action or method name
  API action inputs or parameters: A value
  PropertySelector: Response object
  DesiredValues:
    - Desired property value
```

JSON

```
{
  "action": "aws:waitForAwsResourceProperty",
  "inputs": {
    "Service": "The official namespace of the service",
    "Api": "The API action or method name",
    "API action inputs or parameters": "A value",
    "PropertySelector": "Response object",
    "DesiredValues": [
      "Desired property value"
    ]
  }
}
```

Service

The AWS service namespace that contains the API action that you want to run. For example, the namespace for Systems Manager is `ssm`. The namespace for Amazon EC2 is `ec2`. You can view a list of supported AWS service namespaces in the Available Services section of the AWS CLI Command Reference.

Type: String

Required: Yes

Api

The name of the API action that you want to run. You can view the API actions (also called methods) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all API actions (methods) for Amazon RDS are listed on the following page: Amazon RDS methods.

Type: String

Required: Yes

API action inputs

One or more API action inputs. You can view the available inputs (also called parameters) by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all methods for Amazon RDS are listed on the following page: Amazon RDS methods. Choose the describe_db_instances method and scroll down to see the available parameters, such as `DBInstanceIdentifier`, `Name`, and `Values`.

YAML

```
inputs:
  Service: The official namespace of the service
```
Working with Automation documents (Playbooks)

**Api:** The API action name
**API Input 1:** A value
**API Input 2:** A value
**API Input 3:** A value

**JSON**

```
"inputs":{
  "Service":"The official namespace of the service",
  "Api":"The API action name",
  "API input 1":"A value",
  "API Input 2":"A value",
  "API Input 3":"A value"
}
```

**Type:** Determined by chosen API action

**Required:** Yes

**PropertySelector**

The JSONPath to a specific attribute in the response object. You can view the response objects by choosing a service in the left navigation on the following Services Reference page. Choose a method in the Client section for the service that you want to invoke. For example, all methods for Amazon RDS are listed on the following page: Amazon RDS methods. Choose the describe_db_instances method and scroll down to the Response Structure section. DBInstances is listed as a response object.

**Type:** Integer, Boolean, String, StringList, StringMap, or MapList

**Required:** Yes

**DesiredValues**

The expected status or state on which to continue the Automation workflow.

**Type:** Varies

**Required:** Yes

---

**Systems Manager Automation documents reference**

To help you get started quickly, Systems Manager provides pre-defined Automation documents, or playbooks. These documents are maintained by Amazon Web Services and AWS Support. The Actions Reference describes the actions that you can specify in an Automation document. The Automation Document Details Reference describes each of the predefined Automation documents provided by AWS Systems Manager and AWS Support.

**Important**

If you run an automation workflow that invokes other services by using an AWS Identity and Access Management (IAM) service role, be aware that the service role must be configured with permission to invoke those services. This requirement applies to all AWS Automation documents (AWS-* documents) such as the AWS-ConfigureS3BucketLogging, AWS-CreateDynamoDBBackup, and AWS-RevertEC2Instance documents, to name a few. This requirement also applies to any custom Automation documents you create that invoke other AWS services by using actions that call other services. For example, if you use the aws:executeAwsApi, aws:createStack, or aws:copyImage actions, then you must configure the service role with permission to invoke those services. You can enable permissions to other AWS services by adding an IAM inline policy to the role. For more information, see (Optional) add an Automation inline policy to invoke other AWS services (p. 284).
Working with Automation documents (Playbooks)

Topics
- Automation system variables (p. 435)
- Systems Manager Automation document details reference (p. 443)

Automation system variables

Systems Manager Automation documents use the following variables. For an example of how these variables are used, view the JSON source of the AWS-UpdateWindowsAmi document.

To view the JSON source of the AWS-UpdateWindowsAmi document

2. In the navigation pane, choose Documents.
3. In the document list, use either the Search bar or the numbers to the right of the Search bar to choose the document AWS-UpdateWindowsAmi.
4. Choose the Content tab.

System Variables

Automation documents currently support the following system variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>global:ACCOUNT_ID</td>
<td>The AWS account ID of the AWS Identity and Access Management (IAM) user or role in which Automation runs.</td>
</tr>
<tr>
<td>global:DATE</td>
<td>The date (at execution time) in the format yyyy-MM-dd.</td>
</tr>
<tr>
<td>global:DATE_TIME</td>
<td>The date and time (at execution time) in the format yyyy-MM-dd_HH.mm.ss.</td>
</tr>
<tr>
<td>global:REGION</td>
<td>The Region that the document is run in. For example, us-east-2.</td>
</tr>
</tbody>
</table>

Automation Variables

Automation documents currently support the following automation variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>automation:EXECUTION_ID</td>
<td>The unique identifier assigned to the current automation execution. For example, 1a2b3c-1a2b3c-1a2b3c-1a2b3c1a2b3c1a2b3c1a2b3c.</td>
</tr>
</tbody>
</table>

Topics
- Terminology (p. 436)
- Supported scenarios (p. 438)
- Unsupported scenarios (p. 440)
## Terminology

The following terms describe how variables and parameters are resolved.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant ARN</td>
<td>A valid ARN without variables</td>
<td>arn:aws:iam::123456789012:role/roleName</td>
</tr>
</tbody>
</table>
| Document parameter | A parameter defined at the document level for an Automation document (for example, instanceId). The parameter is used in a basic string replace. Its value is supplied at Start Execution time. | {  
  "description": "Create Image Demo",
  "version": "0.3",
  "assumeRole": "Your_Automation_Assume_Role_ARN",
  "parameters": {
    "instanceId": {
      "type": "String",
      "description": "Instance to create image from"
    }
  }
}                                                                                                                                                    |
| System variable    | A general variable substituted into the document when any part of the document is evaluated.                                      | "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}}"
  }
}                                                                                                                                                    |
| Automation variable| A variable relating to the automation execution substituted into the document when any part of the document is evaluated.         | {  
  "name": "runFixedCmds",
  "action": "aws:runCommand",
  "maxAttempts": 1,
  "onFailure": "Continue",
  "inputs": {
    "DocumentName": "AWS-RunPowerShellScript",
    "InstanceId": [{LaunchInstance.InstanceIds}]
  }
}                                                                                                                                                   |
### "commands": [ "dir",
"date",
""{{outputFormat}}" -f 
"left","right","{{global:DATE}}","{{automation:EXECUTION_ID}}"
] } } } }

### SSM Parameter

A variable defined within Parameter Store. It cannot be directly referenced in step input. Permissions might be required to access the parameter.

```json
description: Launch new Windows test instance
schemaVersion: '0.3'
assumeRole: 
'{{AutomationAssumeRole}}'
parameters:
  AutomationAssumeRole:
    type: String
    default: ''
    description: >-
      (Required) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to execute this document.
  LatestAmi:
    type: String
    default: >-
      {{ssm:/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-Base}}
    description: The latest Windows Server 2016 AMI queried from the public parameter.
mainSteps:
- name: launchInstance
  action: 'aws:runInstances'
  maxAttempts: 3
  timeoutSeconds: 1200
  onFailure: Abort
  inputs:
    ImageId:
      '{LatestAmi}'
...```
## Supported scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comments</th>
<th>Example</th>
</tr>
</thead>
</table>
| Constant ARN assumeRole at creation.               | An authorization check is performed to verify that the calling user is permitted to pass the given assumeRole. | ```json
{
  "description": "Test all Automation resolvable parameters",
  "schemaVersion": "0.3",
  "assumeRole": "arn:aws:iam::123456789012:role/roleName",
  "parameters": {
    ...
``` |
| Document parameter supplied for assumeRole at execution. | Must be defined in the parameter list of the document.                   | ```json
{
  "description": "Test all Automation resolvable parameters",
  "schemaVersion": "0.3",
  "assumeRole": "{{dynamicARN}}",
  "parameters": {
    ...
``` |
| Value supplied for document parameter at start.    | Customer supplies the value to use for a parameter. Any execution inputs supplied at start time need to be defined in the parameter list of the document. | ```json
...
  "parameters": {
    "amiId": {
      "type": "String",
      "default": "ami-12345678",
      "description": "list of commands to run as part of first step"
    },
    ...
``` |
| SSM Parameter referenced within document content.   | The variable exists within the customer's account, or is a publicly accessibly parameter, and the assumeRole for the document has access to the variable. A check is performed at create time to confirm the assumeRole has access. The parameter cannot be directly referenced in step input. | ```json
...
  parameters:
  LatestAmi:
    type: String
    default: >-
      {{ssm:/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-Base}}
    description: The latest Windows Server 2016 AMI queried from the public parameter.
  mainSteps:
    - name: launchInstance
      action:
        'aws:runInstances'
      maxAttempts: 3
``` |
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comments</th>
<th>Example</th>
</tr>
</thead>
</table>
| System variable referenced within step definition | A system variable is substituted into the document at execution time. The value injected into the document is relative to when the substitution occurs. That is, the value of a time variable injected at step 1 is different from the value injected at step 3 because of the time it takes to run the steps between. System variables do not need to be set in the parameter list of the document. | `timeoutSeconds: 1200` `onFailure: Abort` `inputs:` `ImageId:` `'{{LatestAmi}}'` `...`

| Automation variable referenced within step definition | Automation variables do not need to be set in the parameter list of the document. The only supported Automation variable is **automation:EXECUTION_ID**. | `...` `"mainSteps": [` `{` "name": "invokeLambdaFunction", "action": "aws:invokeLambdaFunction", "maxAttempts": 1, "onFailure": "Continue", "inputs": { "FunctionName": "Hello-World-LambdaFunction", "Payload": `{ "executionId": "{{automation:EXECUTION_ID}}" }" ` } `], ...`
### Working with Automation documents (Playbooks)

#### Scenario | Comments |
--- | --- |
Refer to output from previous step within next step definition. | This is parameter redirection. The output of a previous step is referenced using the syntax `{{stepName.OutputName}}`. This syntax cannot be used by the customer for document parameters. This is resolved at the time of execution for the referring step. The parameter is not listed in the parameters of the document. |

```
...  
"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"
  }
]

...  
```

#### Unsupported scenarios

#### Scenario | Comment | Example |
--- | --- | --- |
SSM Parameter supplied for assumeRole at create | Not supported. | ...

```
...  
{  
  "description": "Test all Automation resolvable parameters",
  "schemaVersion": "0.3",
  "assumeRole": "{{ssm:administratorRoleARN}}",
  "parameters": {
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  "environment": {
    "key": "value"
  }
  }  
}

...  
```

SSM Parameter directly referenced in step input. | Returns InvalidDocumentContent exception at create time. | ...

```
...  
mainSteps:  
- name: launchInstance  
```

---

440
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Comment</th>
<th>Example</th>
</tr>
</thead>
</table>
| Variable step definition | The definition of a step in the document is constructed by variables. | action:  
'aws:runInstances'  
maxAttempts: 3  
timeoutSeconds: 1200  
onFailure: Abort  
inputs:  
  ImageId: '{ssm/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-Base}'}  
... |
| Cross referencing document parameters | The user supplies an input parameter at start time, which is a reference to another parameter in the document. | ...  
"mainSteps": [  
  {  
    "name":  
    "LaunchInstance",  
    "action":  
    "aws:runInstances",  
    "{{attemptModel}}": 1,  
    "onFailure":  
    "Continue",  
    "inputs": {  
      "ImageId":  
      "ami-12345678",  
      "MinInstanceCount":  
      1,  
      "MaxInstanceCount":  
      2  
    }  
  }  
...  
User supplies input :  
  { "attemptModel" : "minAttempts" }  
...  
"parameters": {  
  "amiId": {  
    "type": "String",  
    "default":  
    "ami-7f2e6015",  
    "description": "list of commands to run as part of first step"  
  },  
  "alternateAmiId": {  
    "type": "String",  
    "description": "The alternate AMI to try if this first fails".  
  }  
}  
"default" : "{amiId}"
... |
### Scenario
- Multi-level expansion

### Comment
The document defines a variable that evaluates to the name of a variable. This sits within the variable delimiters (that is `{{}}`) and is expanded to the value of that variable/parameter.

### Example
```
...  
  "parameters": { 
    "firstParameter": { 
      "type": "String", 
      "default": "param2", 
      "description": "The parameter to reference" 
    }, 
    "secondParameter": { 
      "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": [ 
          "{{ firstParameter }}"] 
      } 
    } 
  }]
...
```

Note: The customer intention here would be to run a command of "echo {Hello world}"
Scenario | Comment | Example
--- | --- | ---
Referencing output from a document step that is a different variable type | The user references the output from a preceding document step within a subsequent step. The output is a variable type that does not meet the requirements of the action in the subsequent step. | ...
mainSteps:
- name: getImageId
  action: aws:executeAwsApi
  inputs:
    Service: ec2
    Api: DescribeImages
    Filters:
      - Name: "name"
        Values:
          - "{{ImageName}}"
  outputs:
    - Name: ImageIdList
      Selector: "$.Images"
      Type: "StringList"
- name: copyMyImages
  action: aws:copyImage
  maxAttempts: 3
  onFailure: Abort
  inputs:
    SourceImageId: {{getImageId.ImageIdList}}
    SourceRegion: ap-northeast-2
    ImageName: Encrypted
    Copies of LAMP base AMI in ap-northeast-2
    Encrypted: true
...  
Note: You must provide the type required by the Automation action. In this case, aws:copyImage requires a "String" type variable but the preceding step outputs a "StringList" type variable.

Systems Manager Automation document details reference

This section includes topics that describe each of the Systems Manager Automation documents that are owned by AWS and AWS Support. Each page provides an explanation of the required and optional parameters you can specify when using the document. Each page also lists the steps in the document and the output of the execution, if any.

This section does not include a separate page for documents that require approval such as the AWS-CreateManagedLinuxInstanceWithApproval or AWS-StopEC2InstanceWithApproval document. Any document name that includes WithApproval, means the document includes the aws:approve – Pause an execution for manual approval (p. 392) action. This action temporarily pauses an Automation execution until designated principals either approve or reject the action. After the required number of approvals is reached, the Automation execution resumes.

For information about running Automation documents, see Running a simple Automation workflow (p. 286). For information about running Automation documents on multiple targets, see Running Automation workflows that use targets and rate controls (p. 299).
View Automation document content

You can view the content for Automation documents in the Systems Manager console.

To view Automation document content

2. In the navigation pane, choose Documents.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Documents in the navigation pane.
3. Choose a document, and then choose View details.
4. Choose the Content tab.

Topics

- AWSSupport-ActivateWindowsWithAmazonLicense (p. 445)
- AWS-ASGEnterStandby (p. 447)
- AWS-ASGExitStandby (p. 448)
- AWS-AttachEBSVolume (p. 449)
- AWS-AttachIAMToInstance (p. 449)
- AWS-ConfigureCloudWatchOnEC2Instance (p. 451)
- AWS-ConfigureS3BucketLogging (p. 452)
- AWS-ConfigureS3BucketVersioning (p. 453)
- AWS-CopySnapshot (p. 454)
- AWSEC2-CloneInstanceAndUpgradeWindows (p. 455)
- AWSEC2-CloneInstanceAndUpgradeSQLServer (p. 457)
- AWS-CreateDynamoDBBackup (p. 459)
- AWS-CreateImage (p. 460)
- AWS-CreateJiraIssue (p. 460)
- AWS-CreateManagedLinuxInstance (p. 462)
- AWS-CreateManagedWindowsInstance (p. 463)
- AWS-CreateRdsSnapshot (p. 465)
- AWS-CreateServiceNowIncident (p. 466)
- AWS-CreateSnapshot (p. 468)
- AWS-DeleteCloudFormationStack (p. 468)
- AWS-DeleteDynamoDBBackup (p. 469)
- AWS-DeleteDynamoDBTableBackups (p. 469)
- AWS-DeleteEBSVolumeSnapshots (p. 470)
- AWS-DeleteImage (p. 471)
- AWS-DeleteSnapshot (p. 472)
- AWS-DetachEBSVolume (p. 472)
- AWS-DisablePublicAccessForSecurityGroup (p. 473)
- AWS-DisableS3BucketPublicReadWrite (p. 474)
- AWS-EnableCloudTrail (p. 475)
AWSSupport-ActivateWindowsWithAmazonLicense

Description

The AWSSupport-ActivateWindowsWithAmazonLicense automation document activates an EC2 instance for Windows Server with a license provided by Amazon. The automation verifies and configures required key management service operating system settings and attempts activation. This includes operating
system routes to Amazon's key management servers, and key management service operating system settings. Setting the AllowOffline parameter to True allows the automation to successfully target instances that are not managed by AWS Systems Manager, but requires a stop and start of the instance.

**Note**
This document cannot be used on Bring Your Own License (BYOL) Windows Server instances. For information about using your own license, see Microsoft Licensing on AWS.

**Run this Automation (console)**

**Document Type**
Automation

**Owner**
Amazon

**Platforms**
Windows

**Parameters**

- **InstanceId**
  
  **Type:** String
  
  **Description:** (Required) ID of your managed EC2 instance for Windows Server.

- **ForceActivation**
  
  **Type:** String
  
  **Allowed values:** True, False
  
  **Default:** False
  
  **Description:** (Optional) Set it to True if you want to proceed even if Windows is already activated.

- **AllowOffline**
  
  **Type:** String
  
  **Allowed values:** True, False
  
  **Default:** False
  
  **Description:** (Optional) Set it to True if you allow an offline Windows activation remediation in case the online troubleshooting fails, or if the provided instance is not a managed instance.

**Important**
The offline method requires that the provided EC2 instance be stopped and then started. Data stored in instance store volumes will be lost. The public IP address will change if you are not using an Elastic IP.

- **SubnetId**
  
  **Type:** String
  
  **Default:** CreateNewVPC
  
  **Description:** (Optional) Offline only - The subnet ID for the EC2Rescue instance used to perform the offline troubleshooting. Use SelectedInstanceSubnet to use the same subnet as your instance, or
CreateNewVPC to create a new VPC. IMPORTANT: The subnet must be in the same Availability Zone as Instanceld, and it must allow access to the SSM endpoints.

- AutomationAssumeRole

  Type: String

  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

**Required IAM Permissions**

It is recommended that the EC2 instance receiving the command has an IAM role with the AmazonSSMMangedInstanceCore Amazon managed policy attached. You must have at least ssm:ExecuteAutomation and ssm:SendCommand to run the automation and send the command to the instance, plus ssm:GetAutomationExecution to be able to read the automation output. For the offline remediation, see the permissions needed by AWSSupport-StartEC2RescueWorkflow.

**Document Steps**

1. aws:assertAwsResourceProperty - Check the provided instance's platform is Windows.
2. aws:assertAwsResourceProperty - Confirm the provided instance is a managed instance
   a. (Online activation fix) If the input instance is a managed instance, then run aws:runCommand to run the PowerShell script to attempt to fix Windows activation.
   b. (Offline activation fix) If the input instance is not a managed instance:
      i. aws:assertAwsResourceProperty - Verifies the AllowOffline flag is set to True. If so, the offline fix starts, otherwise the workflow ends.
      ii. aws:executeAutomation - Invoke AWSSupport-StartEC2RescueWorkflow with the Windows activation offline fix script. The script leverages EC2Config or EC2Launch depending on the OS version.
      iii. aws:executeAwsApi - Read the result from AWSSupport-StartEC2RescueWorkflow.

**Outputs**

activateWindows.Output
getActivateWindowsOfflineResult.Output

**AWS-ASGEnterStandby**

**Description**

Change the standby state of an EC2 instance in an Auto Scaling group.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux
Parameters

- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- InstanceId
  
  Type: String
  
  Description: (Required) ID of an EC2 instance for which you want to change the standby state within an Auto Scaling group.

- LambdaRoleArn
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.

AWS-ASGExitStandby

Description

Change the standby state of an EC2 instance in an Auto Scaling group.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- InstanceId
  
  Type: String
  
  Description: (Required) ID of an EC2 instance for which you want to change the standby state within an Auto Scaling group.

- LambdaRoleArn
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.
Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.

AWS-AttachEBSVolume

Description
Attach an Amazon Elastic Block Store (Amazon EBS) volume to an EC2 instance.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

• AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

• Device
  Type: String
  Description: (Required) The device name (for example, /dev/sdh or xvdh ).

• InstanceId
  Type: String
  Description: (Required) The ID of the instance where you want to attach the volume.

• VolumeId
  Type: String
  Description: (Required) The ID of the Amazon EBS volume. The volume and instance must be in the same Availability Zone.

AWS-AttachIAMToInstance

Description
Attach an AWS Identity and Access Management (IAM) role to a managed instance.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- InstanceId
  Type: String
  Description: (Required) The ID of the instance on which you want to assign an IAM role.

- RoleName
  Type: String
  Description: (Required) The IAM role name to add to the managed instance.

- ForceReplace
  Type: Boolean
  Description: (Optional) Flag to specify whether to replace the existing IAM profile or not.
  Default: true

Document Steps

1. aws:executeAwsApi - DescribeInstanceProfile - Find the IAM instance profile attached to the EC2 instance.
2. aws:branch - CheckInstanceProfileAssociations - Check the IAM instance profile attached to the EC2 instance.
   a. If an IAM instance profile is attached and ForceReplace is set to true:
      i. aws:executeAwsApi - DisassociateIamInstanceProfile - Disassociate the IAM instance profile from the EC2 instance.
   b. aws:executeAwsApi - ListInstanceProfilesForRole - List instance profiles for the IAM role provided.
   c. aws:branch - CheckInstanceProfileCreated - Check if the IAM role provided has an associated instance profile.
      i. If the IAM role has an associated instance profile:
         A. aws:executeAwsApi - AttachIAMProfileToInstance - Attach the IAM instance profile role to the EC2 instance.
      i. If the IAM role does not have an associated instance profile:
         A. aws:executeAwsApi - CreateInstanceProfileForRole - Create an instance profile role for the specified IAM role.
         B. aws:executeAwsApi - AddRoleToInstanceProfile - Attach the instance profile role to the specified IAM role.
C. aws:executeAwsApi - GetInstanceProfile - Get the instance profile data for the specified IAM role.

D. aws:executeAwsApi - AttachIAMPProfileToInstanceWithRetry - Attach the IAM instance profile role to the EC2 instance.

**Outputs**

AttachIAMPProfileToInstanceWithRetry.AssociationId

GetInstanceProfile.InstanceProfileName

GetInstanceProfile.InstanceProfileArn

AttachIAMPProfileToInstance.AssociationId

ListInstanceProfilesForRole.InstanceProfileName

ListInstanceProfilesForRole.InstanceProfileArn

**AWS-ConfigureCloudWatchOnEC2Instance**

**Description**

Enable or disable Amazon CloudWatch monitoring on managed instances.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **AutomationAssumeRole**

  Type: String

  Description: (Optional) The Amazon Resource Name (ARN) of the role that allows Systems Manager Automation to perform the actions on your behalf.

- **InstanceId**

  Type: String

  Description: (Required) The ID of the Amazon EC2 instance on which you want to enable CloudWatch monitoring.

- **properties**

  Type: String

  Description: (Optional) This parameter is not supported. It is listed here for backwards compatibility.
• status
  Allowed values: Enabled | Disabled
  Description: (Optional) Specifies whether to enable or disable CloudWatch.
  Default: Enabled

**Document Steps**

configureCloudWatch - Configures CloudWatch on the Amazon EC2 instance with the given status.

**Outputs**

The automation execution has no output.

**AWS-ConfigureS3BucketLogging**

**Description**

Enable logging on an Amazon Simple Storage Service (Amazon S3) bucket.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

• AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

• BucketName
  Type: String
  Description: (Required) The name of the Amazon S3 Bucket for which you want to configure logging.

• GrantedPermission
  Type: String
  Allowed values: FULL_CONTROL,READ,WRITE
  Description: (Required) Logging permissions assigned to the grantee for the bucket.

• GranteeEmailAddress
  Type: String
(Optional) Email address of the grantee.

- GranteeId
  
  Type: String
  
  Description: (Optional) The canonical user ID of the grantee.

- GranteeType
  
  Type: String
  
  Allowed values: CanonicalUser, AmazonCustomerByEmail, Group
  
  Description: (Required) Type of grantee.

- GranteeUri
  
  Type: String
  
  Description: (Optional) URI of the grantee group.

- TargetBucket
  
  Type: String
  
  Description: (Required) Specifies the bucket where you want Amazon S3 to store server access logs. You can have your logs delivered to any bucket that you own. You can also configure multiple buckets to deliver their logs to the same target bucket. In this case you should choose a different TargetPrefix for each source bucket so that the delivered log files can be distinguished by key.

- TargetPrefix
  
  Type: String
  
  Default: /
  
  Description: (Optional) Specifies a prefix for the keys under which the log files will be stored.

**AWS-ConfigureS3BucketVersioning**

**Description**

Configure versioning for an Amazon Simple Storage Service (Amazon S3) bucket.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- AutomationAssumeRole
Type: String

Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- BucketName
  Type: String

  Description: (Required) The name of the S3 Bucket whose encryption configuration will be managed.

- VersioningState
  Type: String

  Allowed values: Enabled, Suspended

  Default: Enabled

  Description: (Optional) Applied to the VersioningConfiguration.Status. When set to 'Enabled', this process enables versioning for the objects in the bucket, all objects added to the bucket receive a unique version ID. When set to 'Suspended', this process disables versioning for the objects in the bucket. All objects added to the bucket receive the version ID null.

AWS-CopySnapshot

Description

Copies a point-in-time snapshot of an Amazon Elastic Block Store (Amazon EBS) volume. You can copy the snapshot within the same AWS Region or from one Region to another. Copies of encrypted Amazon EBS snapshots remain encrypted. Copies of unencrypted snapshots remain unencrypted. To copy an encrypted snapshot that was shared from another account, you must have permissions for the AWS KMS customer master key (CMK) used to encrypt the snapshot. Snapshots created by copying another snapshot have an arbitrary volume ID that should not be used for any purpose.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

- AutomationAssumeRole
  Type: String

  Description: (Optional) The Amazon Resource Name (ARN) of the role that allows Systems Manager Automation to perform the actions on your behalf.

- Description
  Type: String
Description: (Optional) A description for the Amazon EBS snapshot.

- **SnapshotId**
  
  Type: String

  Description: (Required) The ID of the Amazon EBS snapshot to copy.

- **SourceRegion**
  
  Type: String

  Description: (Required) The Region where the source snapshot currently exists.

**Document Steps**

copySnapshot - Copies a snapshot of an Amazon EBS volume.

**Outputs**

copySnapshot.SnapshotId - The ID of the new snapshot.

**AWSEC2-CloneInstanceAndUpgradeWindows**

**Description**

Create an Amazon Machine Image (AMI) from a Windows Server 2008 R2, 2012 R2, or 2016 instance, and then upgrade the AMI to Windows Server 2012 R2, 2016, or 2019. The supported upgrade paths are as follows.


To upgrade your instance from Windows Server 2008 R2 to Windows Server 2016 or 2019, the Automation document performs two steps. The Windows Server 2008 R2 instance is upgraded to Windows Server 2012 R2. Then the Windows Server 2012 R2 instance is upgraded to the target version (Windows Server 2016 or 2019).

The upgrade operation is a multi-step process that can take 2 hours to complete. The Automation creates an AMI from the instance and then launches a temporary instance from the newly created AMI in the `SubnetId` that you specify. The security groups associated with your original instance are applied to the temporary instance. The Automation workflow then performs an in-place upgrade to the `TargetWindowsVersion` on the temporary instance. To upgrade your Windows Server 2008 R2 instance to Windows Server 2016 or 2019, an in-place upgrade is performed twice because directly upgrading Windows Server 2008 R2 to Windows Server 2016 or 2019 is not supported. The workflow also updates or installs the AWS drivers required by the temporary instance. After the upgrade, the workflow creates a new AMI from the temporary instance and then terminates the temporary instance.

You can test application functionality by launching a test instance from the upgraded AMI in your VPC. After you finish testing, and before you perform another upgrade, schedule application downtime before completely switching over to the upgraded AMI.

**Run this Automation (console)**

**Document Type**

---

455
Automation

Owner
Amazon

Platforms
Windows Server 2008 R2, 2012 R2 and 2016 Standard and Datacenter editions

Prerequisites

- Verify that SSM Agent is installed on your instance. For more information, see Installing and configuring SSM Agent on Windows Server instances (p. 65).
- For instances that are joined to a Microsoft Active Directory domain, we recommend specifying a SubnetId that does not have connectivity to your domain controllers to help avoid hostname conflicts.
- The SubnetId specified must be a public subnet with the auto-assign public IPv4 address set to true. For more information, see Modifying the Public IPv4 Addressing Attribute for Your Subnet in the Amazon VPC User Guide.
- This Automation works only with Windows Server 2008 R2, 2012 R2, and 2016 instances.
- This Automation works only on instances with an unencrypted EBS root volume. If the specified instance has an encrypted root volume, the Automation workflow fails.
- Configure the Windows Server instance with an AWS Identity and Access Management (IAM) instance profile that provides the requisite permissions for Systems Manager. For more information, see Create an IAM instance profile for Systems Manager (p. 30).
- Verify that the instance has 20 GB of free disk space in the boot disk.
- If the instance does not use an AWS-provided Windows license, then specify an Amazon EBS snapshot ID that includes Windows Server 2012 R2 installation media. To do this:
  - Verify that the EC2 instance is running Windows Server 2012 or later.
  - Create a 6 GB EBS volume in the same Availability Zone where the instance is running. Attach the volume to the instance. Mount it, for example, as drive D.
  - Right-click the ISO and mount it to an instance as, for example, drive E.
  - Copy the content of the ISO from drive E:\ to drive D:\
  - Create an EBS snapshot of the 6 GB volume created in step 2 above.

Limitations
This Automation doesn't support upgrading Windows domain controllers, clusters, or Windows desktop operating systems. This Automation also doesn't support EC2 instances for Windows Server with the following roles installed.

- Remote Desktop Session Host (RDSH)
- Remote Desktop Connection Broker (RDCB)
- Remote Desktop Virtualization Host (RDVH)
- Remote Desktop Web Access (RDWA)

Parameters

- InstanceId
  Type: String
  Description: (Required) The instance running Windows Server 2008 R2 or 2012 R2.
• iamInstanceProfile
  Type: String
  Description: (Required) The name of the IAM instance profile that enables Systems Manager to manage the instance.

• SubnetId
  Type: String
  Description: (Required) Provide a subnet for the upgrade process. Verify that the subnet has outbound connectivity to AWS services, Amazon S3, and Microsoft (to download patches).

• TargetWindowsVersion
  Type: String
  Description: (Required) Select the target Windows version.
  Default: 2012R2

• BYOLWindowsMediaSnapshotId
  Type: String
  Description: (Optional) The ID of the Amazon EBS snapshot to copy that includes Windows Server 2012R2 installation media. Required only if you are upgrading a BYOL instance.

• KeepPreUpgradeImageBackUp
  Type: String
  Description: (Optional) If set True, the Automation doesn't delete the AMI created from the instance before the upgrade. If set to True, then you must delete the AMI. By default, the AMI is deleted.

• RebootInstanceBeforeTakingImage
  Type: String
  Description: (Optional) If set True, the Automation reboots the instance before creating a pre-upgrade AMI. By default, the Automation doesn't reboot before upgrade.

**AWSEC2-CloneInstanceAndUpgradeSQLServer**

**Description**

Create an AMI from an EC2 instance for Windows Server running SQL Server 2008 (or later), and then upgrade the AMI to SQL Server 2016. The upgrade is a multi-step process that can take 2 hours to complete. The Automation creates the AMI from the instance, and then launches the new AMI in the subnet that you provide. The Automation then performs an in-place upgrade of SQL Server 2008 (or later) to SQL Server 2016. After the upgrade, the Automation creates a new AMI before terminating the upgraded instance.

You can test application functionality by launching the new AMI in your VPC. After you finish testing, and before you perform another upgrade, schedule application downtime before completely switching over to the upgraded instance.

**Note**

If you want to modify the computer name of the EC2 instance launched from the new AMI, see [Rename a Computer that Hosts a Stand-Alone Instance of SQL Server](#).

Run this Automation (console)
Document Type

Automation

Owner

Amazon

Platforms

Windows

Prerequisites

- The EC2 instance must use a version of Windows Server that is Windows Server 2008 R2 (or later) and SQL Server 2008 (or later).
- Verify that SSM Agent is installed on your instance. For more information, see Installing and configuring SSM Agent on Windows Server instances (p. 65).
- Configure the instance to use an AWS Identity and Access Management (IAM) instance profile role. For more information, see Create an IAM instance profile for Systems Manager (p. 30).
- Verify that the instance has 20 GB of free disk space in the instance boot disk.
- For instances that use a Bring Your Own License (BYOL) SQL Server version, the following additional prerequisites apply:
  - Provide an EBS snapshot ID that includes SQL Server 2016 installation media. To do this:
    1. Verify that the EC2 instance is running Windows Server 2008 R2 or later.
    2. Create a 6 GB EBS volume in the same Availability Zone where the instance is running. Attach the volume to the instance. Mount it, for example, as drive D.
    3. Right-click the ISO and mount it to an instance as, for example, drive E.
    4. Copy the content of the ISO from drive E:\ to drive D:\
    5. Create an EBS snapshot of the 6 GB volume created in step 2.

Limitations

- The upgrade can only be performed on a SQL Server using Windows authentication.
- Verify that no security patch updates are pending on the instances. Open Control Panel, then choose Check for updates.
- SQL Server deployments in HA and mirroring mode are not supported.

Parameters

- InstanceId
  Type: String
  Description: (Required) The instance running Windows Server 2008 R2 (or later) and SQL Server 2008 (or later).
- IamInstanceProfile
  Type: String
  Description: (Required) The IAM instance profile.
- SubnetId
  Type: String
AWS Systems Manager User Guide
Working with Automation documents (Playbooks)

Description: (Required) Provide a subnet for the upgrade process. Verify that the subnet has outbound connectivity to AWS services, Amazon S3, and Microsoft (to download patches).
- SQLServerSnapshotId
  Type: String
  Description: (Conditional) Snapshot ID for SQL Server 2016 installation media. This parameter is required for instances that use a BYOL SQL Server version. This parameter is optional for SQL Server license-included instances (instances launched using an AWS provided Amazon Machine Image for Windows Server with Microsoft SQL Server).
- KeepPreUpgradeImageBackUp
  Type: String
  Description: (Optional) If set to True, the Automation doesn't delete the AMI created from the instance before the upgrade. If set to True, then you must delete the AMI. By default, the AMI is deleted.
- RebootInstanceBeforeTakingImage
  Type: String
  Description: (Optional) If set to True, the Automation reboots the instance before creating a pre-upgrade AMI. By default, the Automation doesn't reboot before upgrade.

Outputs
AMIId: The ID of the AMI created from the instance that was upgraded to SQL Server 2016

AWS-CreateDynamoDBBackup

Description
Create a backup of an Amazon DynamoDB table.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
This document is not restricted to specific operating system.

Parameters
- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- BackupName
  Type: String
Description: (Required) Name of the backup to create.

- **LambdaAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.

- **TableName**
  
  Type: String
  
  Description: (Required) Name of the DynamoDB table.

**AWS-CreateImage**

**Description**

Create a new Amazon Machine Image (AMI) from an EC2 instance.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

- **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- **InstanceId**
  
  Type: String
  
  Description: (Required) The ID of the EC2 instance.

- **NoReboot**
  
  Type: Boolean
  
  Description: (Optional) Do not reboot the instance before creating the image.

**AWS-CreateJiraIssue**

**Description**

Create an issue in Jira.

**Run this Automation (console)**
Document Type
Automation

Owner
Amazon

Platform(s)
Windows, Linux

Parameters
- IssueDescription
  Type: String
  Description: (Required) A detailed description of the issue.
- IssueSummary
  Type: String
  Description: (Required) A brief summary of the issue.
- IssueTypeName
  Type: String
  Description: (Required) The name of the type of issue you want to create (for example, Task, Sub-task, Bug, etc.).
- JiraURL
  Type: String
  Description: (Required) The url of the Jira instance.
- JiraUsername
  Type: String
  Description: (Required) The name of the user the issue will be created with.
- ProjectKey
  Type: String
  Description: (Required) The key of the project the issue should be created in.
- SSMParameterName
  Type: String
  Description: (Required) The name of an encrypted SSM Parameter containing the API key or password for the Jira user.
- AssigneeName
  Type: String
  Description: (Optional) The username of the person the issue should be assigned to.
- DueDate
  Type: String
Description: (Optional) The due date for the issue in yyyy-mm-dd format.
- PriorityName
  
  Type: String
  
  Description: (Optional) The name of the priority of the issue.

Document Steps

aws:createStack - Create CloudFormation stack to create Lambda IAM role and function.
aws:invokeLambdaFunction - Invoke Lambda function to create the Jira issue
aws:deleteStack - Delete the CloudFormation stack created.

Outputs

IssueId: ID of the newly created Jira issue

AWS-CreateManagedLinuxInstance

Description

Create an EC2 instance for Linux that is configured for Systems Manager.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

- AmiId
  
  Type: String
  
  Description: (Required) AMI ID to use for launching the instance.
- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf
- GroupName
  
  Type: String
  
  Default: SSMSecurityGroupForLinuxInstances
  
  Description: (Required) Security group name to create.
• InstanceType
  Type: String
  Default: t2.medium
  Description: (Required) Type of instance to launch. Default is t2.medium.
• KeyPairName
  Type: String
  Description: (Required) Key pair to use when creating instance.
• RemoteAccessCidr
  Type: String
  Default: 0.0.0.0/0
  Description: (Required) Creates Security group with port for SSH(Port range 22) open to IPs specified by CIDR (default is 0.0.0.0/0). If the security group already exists it will not be modified and rules will not be changed.
• RoleName
  Type: String
  Default: SSMMangedInstanceProfileRole
  Description: (Required) Role name to create.
• StackName
  Type: String
  Default: CreateManagedInstanceStack{{automation:EXECUTION_ID}}
  Description: (Optional) Specify stack name used by this document
• SubnetId
  Type: String
  Default: Default
  Description: (Required) New instance will be deployed into this subnet or in the default subnet if not specified.
• VpcId
  Type: String
  Default: Default
  Description: (Required) New instance will be deployed into this Amazon Virtual Private Cloud (Amazon VPC) or in the default Amazon VPC if not specified.

AWS-CreateManagedWindowsInstance

Description
Create an EC2 instance for Windows Server that is configured for Systems Manager.

Run this Automation (console)
Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

- AmiId
  Type: String
  Default: {{ssm:/aws/service/ami-windows-latest/Windows_Server-2016-English-Full-Base}}
  Description: (Required) AMI ID to use for launching the instance.

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf

- GroupName
  Type: String
  Default: SSMSecurityGroupForLinuxInstances
  Description: (Required) Security group name to create.

- InstanceType
  Type: String
  Default: t2.medium
  Description: (Required) Type of instance to launch. Default is t2.medium.

- KeyPairName
  Type: String
  Description: (Required) Key pair to use when creating instance.

- RemoteAccessCidr
  Type: String
  Default: 0.0.0.0/0
  Description: (Required) Creates security group with port for RDP (Port range 3389) open to IPs specified by CIDR (default is 0.0.0.0/0). If the security group already exists it will not be modified and rules will not be changed.

- RoleName
  Type: String
  Default: SSMManagedInstanceProfileRole
Description: (Required) Role name to create.

- **StackName**
  
  Type: String
  
  Default: CreateManagedInstanceStack{{automation:EXECUTION_ID}}
  
  Description: (Optional) Specify stack name used by this document

- **SubnetId**
  
  Type: String
  
  Default: Default
  
  Description: (Required) New instance will be deployed into this subnet or in the default subnet if not specified.

- **VpcId**
  
  Type: String
  
  Default: Default
  
  Description: (Required) New instance will be deployed into this Amazon Virtual Private Cloud (Amazon VPC) or in the default Amazon VPC if not specified.

---

**AWS-CreateRdsSnapshot**

**Description**

Create an Amazon Relational Database Service (Amazon RDS) snapshot for an Amazon RDS instance.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **DBInstanceIdentifier**
  
  Type: String
  
  Description: (Required) The DBInstanceId ID of the RDS Instance to create Snapshot from.

- **DBSnapshotIdentifier**
  
  Type: String
  
  Description: (Optional) The DBSnapshotIdentifier ID of the RDS snapshot to create.

- **InstanceTags**
Type: String
Description: (Optional) Tags to create for instance.
- SnapshotTags
  Type: String
  Description: (Optional) Tags to create for snapshot.
- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

Document Steps
createRDSSnapshot – Creates the RDS snapshot and returns the snapshot ID.
verifyRDSSnapshot – Checks that the snapshot created in the previous step exists.

Outputs
createRDSSnapshot.SnapshotId – The ID of the created snapshot.

AWS-CreateServiceNowIncident
Description
This document creates an incident in the ServiceNow incident table.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
- ServiceNowInstanceUsername
  Type: String
  Description: (Required) The name of the user the incident will be created with.
- ServiceNowInstancePassword
  Type: String
  Description: (Required) The name of an encrypted SSM Parameter containing the password for the ServiceNow user.
• ServiceNowInstanceURL
  Type: String
  Description: (Required) The URL of the ServiceNow instance

• ShortDescription
  Type: String
  Description: (Required) A brief description of the incident.

• Description
  Type: String
  Description: (Required) A detailed explanation on the incident.

• Impact
  Type: String
  Description: (Optional) The effect an incident has on business.
  Valid Values: High | Medium | Low
  Default Value: Low

• Category
  Type: String
  Description: (Optional) The category of the incident.
  Valid Values: None | Inquiry/Help | Software | Hardware | Network | Database
  Default Value: None

• Subcategory
  Type: String
  Description: (Optional) The subcategory of the incident.
  Valid Values: None | Antivirus | Email | Internal Application | Operating System | CPU | Disk | Keyboard | Hardware | Memory | Monitor | Mouse | DHCP | DNS | IP Address | VPN | Wireless | DB2 | MS SQL Server | Oracle
  Default Value: None

• AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

**Document Steps**

Push_incident – Pushes the incident information to ServiceNow.

**Outputs**

Push_incident.incidentID – The created incident ID.
AWS-CreateSnapshot

Description
Create a snapshot of an Amazon EBS volume.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- Description
  Type: String
  Description: (Optional) A description for the snapshot
- VolumeId
  Type: String
  Description: (Required) The ID of the volume.

AWS-DeleteCloudFormationStack

Description
Delete an AWS CloudFormation stack.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
- **AutomationAssumeRole**
  
  *Type: String*
  
  *Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.*

- **StackNameOrId**
  
  *Type: String*
  
  *Description: (Required) Name or Unique ID of the CloudFormation stack to be deleted.*

---

**AWS-DeleteDynamoDBBackup**

**Description**

Delete the backup of an Amazon DynamoDB table.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **AutomationAssumeRole**
  
  *Type: String*
  
  *Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.*

- **BackupArn**
  
  *Type: String*
  
  *Description: (Required) ARN of the DynamoDB table backup to delete.*

---

**AWS-DeleteDynamoDBTableBackups**

**Description**

Delete DynamoDB table backups based on retention days or count.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**
Amazon

Platforms

Windows, Linux

Parameters

• AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

• LambdaAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.

• RetentionCount
  Type: String
  Default: 10
  Description: (Optional) The number of backups to retain for the table. If more than the specified number of backup exist, the oldest backups beyond that number are deleted. Either RetentionCount or RetentionDays can be used, not both.

• RetentionDays
  Type: String
  Description: (Optional) The number of days to retain backups for the table. Backups older than the specified number of days are deleted. Either RetentionCount or RetentionDays can be used, not both.

• TableName
  Type: String
  Description: (Required) Name of the DynamoDB table.

AWS-DeleteEBSVolumeSnapshots

Description

Delete a snapshot of an Amazon Elastic Block Store (Amazon EBS) volume.

Note
The AWS Lambda function that runs during this operation has a maximum execution time (timeout) of 60 seconds. If you have a large number of Amazon EBS volume snapshots to delete, the operation might fail with an error message.

Run this Automation (console)

Document Type

Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- LambdaAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.
- RetentionCount
  Type: String
  Default: 10
  Description: (Optional) Number of snapshots to keep for the volume. Either RetentionCount or RetentionDays should be mentioned, not both.
- RetentionDays
  Type: String
  Description: (Optional) Number of days to keep snapshots for the volume. Either RetentionCount or RetentionDays should be mentioned, not both.
- VolumeId
  Type: String
  Description: (Required) The volume identifier to delete snapshots for.

AWS-DeleteImage

Description
Delete an Amazon Machine Image (AMI) and all associated snapshots.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux
Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- ImageId
  Type: String
  Description: (Required) The ID of the AMI.

AWS-DeleteSnapshot

Description
Delete a snapshot of an Amazon EBS volume.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- SnapshotId
  Type: String
  Description: (Required) The ID of the EBS snapshot.

AWS-DetachEBSVolume

Description
Detach an Amazon EBS volume from an EC2 instance.

Run this Automation (console)

Document Type
Automation
Owner
Amazon

Platforms
Windows, Linux

Parameters

- **AutomationAssumeRole**
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- **LambdaAssumeRole**
  Type: String
  Description: (Optional) The ARN of the role assumed by Lambda

- **VolumeId**
  Type: String
  Description: (Required) The ID of the EBS volume. The volume and instance must be within the same Availability Zone

*AWS-DisablePublicAccessForSecurityGroup*

**Description**

This document disables default SSH and RDP ports that are opened to all IP addresses.

**Important**

This document fails with an "InvalidPermission.NotFound" error for security groups that meet both of the following criteria: 1) The security group is located in a non-default VPC; and 2) The inbound rules for the security group don't specify open ports using all four of the following patterns:

- `0.0.0.0/0`
- `::/0`
- `SSH or RDP port + 0.0.0.0/0`
- `SSH or RDP port + ::/0`

If the security group is located in a non-default VPC and, for example, specifies open ports using only the `SSH or RDP port + 0.0.0.0/0` format, then the document fails to run.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon
Platform(s)
Windows, Linux

Parameters

• **GroupId**
  
  Type: String
  
  Description: (Required) The ID of the security group for which the ports should be disabled.

• **IpAddressToBlock**
  
  Type: String
  
  Description: (Optional) Additional IPv4 addresses from which access should be blocked, in the format 1.2.3.4/32.

• **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

Outputs
None

AWS-DisableS3BucketPublicReadWrite

Description
Use Amazon Simple Storage Service (Amazon S3) Block Public Access to disable read and write access for a public S3 bucket. For more information, see Using Amazon S3 Block Public Access in the Amazon Simple Storage Service Developer Guide.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

• **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

• **S3BucketName**
Type: String
Description: (Required) S3 bucket on which you want to restrict access.

**AWS-EnableCloudTrail**

**Description**
Create an AWS CloudTrail trail and configure logging to an S3 bucket.

**Run this Automation (console)**

**Document Type**
Automation

**Owner**
Amazon

**Platform(s)**
Windows, Linux

**Parameters**
- **AutomationAssumeRole**
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- **S3BucketName**
  Type: String
  Description: (Required) Name of the S3 bucket designated for publishing log files.

  **Note**
  The S3 bucket must exist and the bucket policy must grant CloudTrail permission to write to it. For information, see [Amazon S3 Bucket Policy for CloudTrail](#).

- **TrailName**
  Type: String
  Description: (Required) The name of the new trail.

**AWS-EnableS3BucketEncryption**

**Description**
Enable encryption for an Amazon Simple Storage Service (Amazon S3) bucket (encrypt the contents of the bucket).

**Run this Automation (console)**

**Document Type**
Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- BucketName
  Type: String
  Description: (Required) The name of the S3 bucket where you want to encrypt the contents.

- SSEAlgorithm
  Type: String
  Default: AES256
  Description: (Optional) Server-side encryption algorithm to use for the default encryption.

AWS-ExportOpsDataToS3

Description

This document retrieves a list of OpsData summaries in AWS Systems Manager Explorer and exports them to an object in a specified S3 bucket.

Run this Automation (console)
• filters
  Type: String
  Description: (Optional) Filters for the getOpsSummary request.
• syncName
  Type: String
  Description: (Optional) The name of the resource data sync.
• resultAttribute
  Type: String
  Description: (Optional) The result attribute for getOpsSummary request.
• columnFields
  Type: StringList
  Description: (Required) Column fields to write to the output file.
• s3BucketName
  Type: String
  Description: (Required) S3 bucket where you want to download the output file.
• snsTopicArn
  Type: String
  Description: (Required) Amazon Simple Notification Service (Amazon SNS) topic ARN to notify when the download completes.
• snsSuccessMessage
  Type: String
  Description: (Optional) Message to send when document finishes.

**Document Steps**

getOpsSummaryStep – Retrieves up to 5,000 ops summaries to export in a CSV file now.

**Outputs**

OpsData object – If the document is executed successfully, you will find the exported OpsData object in your target S3 bucket.

**AWSSupport-ExecuteEC2Rescue**

**Description**

This document will use the EC2Rescue tool to troubleshoot and where possible repair common connectivity issues with the specified EC2 instance for Linux or Windows Server.

**Run this Automation (console)**

**Document Type**

Automation
Owner
Amazon

Platforms
Windows, Linux

Parameters

• UnreachableInstanceId
  Type: String
  Description: (Required) ID of your unreachable EC2 instance. IMPORTANT: AWS Systems Manager Automation stops this instance, and creates an AMI before attempting any operations. Data stored in instance store volumes will be lost. The public IP address will change if you are not using an Elastic IP.

• SubnetId
  Type: String
  Default: CreateNewVPC
  Description: (Optional) The subnet ID for the EC2Rescue instance. By default, AWS Systems Manager Automation creates a new VPC. Alternatively, UseSelectedInstanceSubnet to use the same subnet as your instance, or specify a custom subnet ID. IMPORTANT: The subnet must be in the same Availability Zone as UnreachableInstanceId, and it must allow access to the SSM endpoints.

• EC2RescueInstanceType
  Type: String
  Allowed values: t2.small, t2.medium, t2.large
  Default: t2.small
  Description: (Required) The EC2 instance type for the EC2Rescue instance. Recommended size: t2.small.

• LogDestination
  Type: String
  Description: (Optional) S3 bucket name in your account where you want to upload the troubleshooting logs. Make sure the bucket policy does not grant unnecessary read/write permissions to parties that do not need access to the collected logs.

• AssumeRole
  Type: String
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use your IAM permissions to run this document.

Required IAM Permissions
You must have at least ssm:ExecuteAutomation and ssm:GetAutomationExecution to be able to read the automation output. For more information about the required permissions see AWSSupport-StartEC2RescueWorkflow

Document Steps
1. **aws:assertAwsResourceProperty** - Assert if the provided instance is Windows
   a. (EC2Rescue for Windows) If the provided instance is a Windows Server instance:
      i. **aws:executeAutomation** - Invoke AWSSupport-StartEC2RescueWorkflow with the EC2Rescue for Windows offline script
      ii. **aws:executeAwsApi** - Retrieve the backup AMI ID from the nested automation
      iii. **aws:executeAwsApi** - Retrieve the EC2Rescue summary from the nested automation
   b. (EC2Rescue for Linux) If the provided instance is a Linux instance:
      i. **aws:executeAutomation** - Invoke AWSSupport-StartEC2RescueWorkflow with the EC2Rescue for Linux offline script
      ii. **aws:executeAwsApi** - Retrieve the backup AMI ID from the nested automation
      iii. **aws:executeAwsApi** - Retrieve the EC2Rescue summary from the nested automation

**Outputs**

- getEC2RescueForWindowsResult.Output
- getWindowsBackupAmi.ImageId
- getEC2RescueForLinuxResult.Output
- getLinuxBackupAmi.ImageId

**AWSSupport-GrantPermissionsToIAMUser**

**Description**

This document grants the specified permissions to an IAM group (new or existing), and adds the existing IAM user to it. Policies you can choose from: Billing or Support. To enable billing access for IAM, remember to also activate IAM user and federated user access to the Billing and Cost Management pages.

**Important**

If you provide an existing IAM group, all current IAM users in the group receive the new permissions.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Parameters**

- **IAMUserName**
  Type: String
  Default: ExampleUser
  Description: (Required) Must be an existing user.
- **IAMGroupName**
  Type: String
Default: ExampleSupportAndBillingGroup

Description: (Required) Can be a new or existing group. Must comply with IAM Entity Name Limits.

- Permissions
  
  Type: String
  
  Allowed values: SupportFullAccess, BillingFullAccess, SupportAndBillingFullAccess
  
  Default: SupportAndBillingFullAccess
  

- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

- LambdaAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role assumed by lambda.

**Required IAM Permissions**

Least privileges depend on how AWSSupport-GrantPermissionsToIAMUser is run.

**Direct execution**

It is recommended you have the **AmazonSSMAutomationRole** Amazon managed policy attached, and the following additional permissions to be able to create the Lambda function and the IAM Role to pass to Lambda:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Action": ["lambda:InvokeFunction", "lambda:CreateFunction", "lambda:DeleteFunction", "lambda:GetFunction"],
         "Resource": "arn:aws:lambda:*:ACCOUNTID:function:AWSSupport-*",
         "Effect": "Allow"
      },
      {
         "Effect": "Allow",
      }
   ]
}
```
Using AutomationAssumeRole and LambdaAssumeRole

The user must have the **ssm:ExecuteAutomation** permissions on the document, and **iam:PassRole** on the IAM roles passed as **AutomationAssumeRole** and **LambdaAssumeRole**. Here are the permissions each IAM role needs:

### AutomationAssumeRole

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lambda:InvokeFunction",
        "lambda:CreateFunction",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource": "arn:aws:lambda::ACCOUNTID:function:AWSSupport-*",
      "Effect": "Allow"
    }
  ]
}
```

### LambdaAssumeRole

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:AttachGroupPolicy",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "iam:PolicyArn": [
            "arn:aws:iam::aws:policy/AWSSupportAccess"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListAccountAliases",
        "iam:GetAccountSummary"
      ],
      "Resource": "*"
    }
  ]
}
```
"Effect" : "Allow",
"Action" : [
    "iam:CreateGroup",
    "iam:AddUserToGroup",
    "iam:ListAttachedGroupPolicies",
    "iam:GetGroup",
    "iam:GetUser"
],
"Resource" : [
    "arn:aws:iam::*:user/*",
    "arn:aws:iam::*:group/*"
],
},
{
  "Effect" : "Allow",
  "Action" : [
    "iam:AttachGroupPolicy"
  ],
  "Resource" : "*",
  "Condition": {
    "ArnEquals": {
      "iam:PolicyArn": [
        "arn:aws:iam::aws:policy/AWSSupportAccess"
      ]
    }
  }
},
{
  "Effect" : "Allow",
  "Action" : [
    "iam:ListAccountAliases",
    "iam:GetAccountSummary"
  ],
  "Resource" : "*"
}
]

Document Steps

1. aws:createStack - Run AWS CloudFormation Template to create a Lambda function.
2. aws:invokeLambdaFunction - Run Lambda to set IAM permissions.
3. aws:deleteStack - Delete CloudFormation Template.

Outputs

configureIAM.Payload

AWSSupport-ManageRDPSettings

Description

The AWSSupport-ManageRDPSettings automation document allows the user to manage common Remote Desktop Protocol (RDP) settings, such as the RDP port and Network Layer Authentication (NLA). By default, the document reads and outputs the values of the settings.

Important

Changes to the RDP settings should be carefully reviewed before running this document.
Document Type
Automation

Owner
Amazon

Platforms
Windows

Parameters

• InstanceId
  Type: String
  Description: (Required) The ID of the managed instance to manage the RDP settings of.

• RDPPortAction
  Type: String
  Allowed values: Check, Modify
  Default: Check
  Description: (Required) An action to apply to the RDP port: Check, Modify.

• RDPPort
  Type: String
  Default: 3389
  Description: (Optional) Specify the new RDP port. Used only when the action is set to Modify. The port number must be between 1025-65535. Note: After the port is changed, the RDP service is restarted.

• NLASettingAction
  Type: String
  Allowed values: Check, Enable, Disable
  Default: Check
  Description: (Required) An action to perform on the NLA setting: Check, Enable, Disable.

• RemoteConnections
  Type: String
  Allowed values: Check, Enable, Disable
  Default: Check
  Description: (Required) An action to perform on the fDenyTSConnections setting: Check, Enable, Disable.

• AutomationAssumeRole
  Type: String
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.
Required IAM Permissions

The EC2 instance receiving the command must have an IAM role with the `AmazonSSMManagedInstanceCore` Amazon managed policy attached. The user must have at least `ssm:SendCommand` to send the command to the instance, plus `ssm:GetCommandInvocation` to be able to read the command output.

Document Steps

```bash
aws:runCommand - Run the PowerShell script to change or check the RDP settings on the target instance.
```

Outputs

```text
manageRDPSettings.Output
```

**AWSSupport-ManageWindowsService**

Description

The AWSSupport-ManageWindowsService automation document enables a user to stop, start, restart, pause, or disable any Windows service on the target instance.

Run this Automation (console)

Document Type

Automation

**Owner**

Amazon

**Platforms**

Windows

**Parameters**

- InstanceId
  
  Type: String
  
  Description: (Required) The ID of the managed instance to manage the RDP settings of.

- WindowsServiceName
  
  Type: String
  
  Description: (Required) A valid Windows service name.

- StartupType
  
  Type: String
  
  Allowed values: Check, Auto, Demand, Disabled, DelayedAutoStart
  
  Default: Check
  
  Description: (Required) A startup type to apply to the Windows service: Auto, Demand (Manual), Disabled, DelayedAutoStart, Check.
• ServiceAction
  Type: String
  Allowed values: Check, Restart, Force-Restart, Start, Stop, Force-Stop, Pause
  Default: Check
  Description: (Required) An action to apply to the Windows service: Restart, Force-Restart, Start, Stop, Force-Stop, Pause, Check. Note: Force-Restart and Force-Stop can be used to restart and to stop a service that has dependent services.

• AutomationAssumeRole
  Type: String
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

Required IAM Permissions

It is recommended that the EC2 instance receiving the command has an IAM role with the AmazonSSMManagedInstanceCore Amazon managed policy attached. The user must have at least ssm:ExecuteAutomation and ssm:SendCommand to run the automation and send the command to the instance, plus ssm:GetAutomationExecution to be able to read the automation output.

Document Steps

aws:runCommand - Run the PowerShell script to apply the desired configuration to the Windows service on the target instance.

Outputs

manageWindowsService.Output

AWS-PatchASGInstance

Description

Patch EC2 instances in an Auto Scaling group.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

• AutomationAssumeRole
  Type: String
Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- **InstanceId**
  
  Type: String
  
  Description: (Required) ID of the instance to patch. Don’t specify an instance ID that is configured to run during a Maintenance Window.

- **LambdaRoleArn**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.

- **WaitForInstance**
  
  Type: String
  
  Default: PT2M
  
  Description: (Optional) Duration the Automation should sleep to allow the instance to come back into service.

- **WaitForReboot**
  
  Type: String
  
  Default: PT5M
  
  Description: (Optional) Duration the Automation should sleep to allow a patched instance to reboot.

**AWS-PatchInstanceWithRollback**

**Description**

Brings EC2 instance into compliance with standing baseline; rolls back root volume on failure.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
• Instanceld
  Type: String
  Description: (Required) EC2 Instanceld to which we apply the patch-baseline.
• LambdaAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.
• ReportS3Bucket
  Type: String
  Description: (Optional) Amazon S3 Bucket destination for the Compliance Report generated during process.

Document Steps

<table>
<thead>
<tr>
<th>Step number</th>
<th>Step name</th>
<th>Automation action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>createDocumentStack</td>
<td>aws:createStack</td>
</tr>
<tr>
<td>2</td>
<td>IdentifyRootVolume</td>
<td>aws:invokeLambdaFunction</td>
</tr>
<tr>
<td>3</td>
<td>PrePatchSnapshot</td>
<td>aws:executeAutomation</td>
</tr>
<tr>
<td>4</td>
<td>installMissingUpdates</td>
<td>aws:runCommand</td>
</tr>
<tr>
<td>5</td>
<td>SleepThruInstallation</td>
<td>aws:invokeLambdaFunction</td>
</tr>
<tr>
<td>6</td>
<td>CheckCompliance</td>
<td>aws:invokeLambdaFunction</td>
</tr>
<tr>
<td>7</td>
<td>SaveComplianceReportToS3</td>
<td>aws:invokeLambdaFunction</td>
</tr>
<tr>
<td>8</td>
<td>ReportSuccessOrFailure</td>
<td>aws:invokeLambdaFunction</td>
</tr>
<tr>
<td>9</td>
<td>DeleteSnapshot</td>
<td>aws:invokeLambdaFunction</td>
</tr>
<tr>
<td>10</td>
<td>createDocumentStack</td>
<td>aws:deleteStack</td>
</tr>
</tbody>
</table>

Outputs

IdentifyRootVolume.Payload
PrePatchSnapshot.Output
SaveComplianceReportToS3.Payload
RestoreFromSnapshot.Payload
CheckCompliance.Payload
AWS-PublishSNSNotification

Description

Publish a notification to Amazon SNS.
Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platform(s)
Windows, Linux

Parameters
• AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
• Message
  Type: String
  Description: (Required) The message to include in the SNS notification.
• TopicArn
  Type: String
  Description: (Required) The ARN of the SNS topic to publish the notification to.

AWS-RebootRDSInstance

Description
The AWS-RebootRDSInstance Automation document reboots an Amazon Relational Database Service (Amazon RDS) DB instance if it is not already rebooting.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
• AutomationAssumeRole
  Type: String
Description: (Optional) The Amazon Resource Name (ARN) of the role that allows Systems Manager Automation to perform the actions on your behalf.

- `InstanceId`
  
  **Type:** String
  
  **Description:** (Required) The ID of the Amazon RDS DB instance that you want to reboot.

**Document Steps**

- **RebootInstance** - Reboots the DB instance if it is not already rebooting.
- **WaitForAvailableState** - Waits for the DB instance to complete the reboot process.

**Outputs**

The automation execution has no outputs.

**AWS-RunCfnLint**

**Description**

This document uses an AWS CloudFormation Linter (cfn-python-lint) to validate YAML and JSON templates against the AWS CloudFormation resource specification. The AWS-RunCfnLint document performs additional checks, such as ensuring that valid values have been entered for resource properties. If validation is not successful, the RunCfnLintAgainstTemplate step fails and the linter tool's output is provided in an error message. This Automation document is using cfn-lint v0.24.4.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **TemplateS3BucketName**
  
  **Type:** String
  
  **Description:** The name of the S3 bucket containing the packer template.
- **TemplateName**
  
  **Type:** String
  
  **Description:** The name, or key, of the template file in the S3 bucket.
- **FormatFlag**
  
  **Type:** String
  
  **Description:** (Optional) Value to pass to the --format parameter to specify the output format.
Valid Values: Default | quiet | parseable | json

Default: Default

- RegionsFlag
  Type: String
  Description: (Optional) Values to pass to the `--regions` parameter to test the template against specified AWS Regions.
  Example: us-east-1,us-west-1

- IgnoreChecksFlag
  Type: String
  Description: (Optional) IDs of rules to pass to the `--ignore-checks` parameter. These rules are not checked.
  Example: E1001,E1003,W7001

- IncludeChecksFlag
  Type: String
  Description: (Optional) IDs of rules to pass to the `--include-checks` parameter. These rules are checked.
  Example: E1001,E1003,W7001

- ConfigureRuleFlag
  Type: String
  Description: (Optional) Configuration options for a rule to pass to the `--configure-rule` parameter.

- InfoFlag
  Type: String
  Description: (Optional) Option for the `--info` parameter. Include the option to enable additional logging information about the template processing.
  Default: False

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

**Document Steps**

RunCfnLintAgainstTemplate – Runs the `cfn-python-lint` tool against the specified AWS CloudFormation template.

**Outputs**

RunCfnLintAgainstTemplate.output – The stdout from the `cfn-python-lint` tool.
AWS-RunPacker

Description

This document uses the HashiCorp Packer tool to validate, fix, or build packer templates that are used to create machine images. This document uses Packer v1.4.4.

Note

If you specify a `vpc_id` value, you must also specify the `subnet_id` value of a public subnet. Unless you modify your subnet’s IPv4 public addressing attribute, you must also set `associate_public_ip_address` to true.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

- TemplateS3BucketName
  
  Type: String
  
  Description: The name of the S3 bucket containing the packer template.

- TemplateFileName
  
  Type: String
  
  Description: The name, or key, of the template file in the S3 bucket.

- Mode
  
  Type: String
  
  Description: The mode, or command, in which to use Packer when validating against the template. Options include Build, Validate, and Fix.

- Force
  
  Type: Boolean
  
  Description: A Packer option to force a builder to run when artifacts from a previous build otherwise prevent a build from running.

- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

Document Steps
RunPackerProcessTemplate – Runs the selected mode against the template using the Packer tool.

**Outputs**

RunPackerProcessTemplate.output – The stdout from the Packer tool.

RunPackerProcessTemplate.fixed_template_key – The name of the template stored in an S3 bucket to use only when running in "Fix" mode.

RunPackerProcessTemplate.s3_bucket – The name of the S3 bucket that contains the fixed template to use only when running in "Fix" mode.

**AWSSupport-ResetAccess**

**Description**

This document will use the EC2Rescue tool on the specified EC2 instance to re-enable password decryption via the EC2 Console (Windows), or to generate and add a new SSH key pair (Linux). If you lost your key pair, this automation will create a password-enabled AMI that you can use to launch a new EC2 instance with a key pair you own (Windows).

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **InstanceId**
  
  Type: String
  
  Description: (Required) ID of the EC2 instance you want to reset access for.

  **Important**
  
  Systems Manager Automation stops this instance, and creates an AMI before attempting any operations. Data stored in instance store volumes will be lost. The public IP address will change if you are not using an Elastic IP.

- **SubnetId**
  
  Type: String
  
  Default: CreateNewVPC
  
  Description: (Optional) The subnet ID for the EC2Rescue instance. By default, Systems Manager Automation creates a new VPC. Alternatively, Use SelectedInstanceSubnet to use the same subnet as your instance, or specify a custom subnet ID.

  **Important**
  
  The subnet must be in the same Availability Zone as Instanceld, and it must allow access to the SSM endpoints.

- **EC2RescueInstanceType**
Type: String

Allowed values: t2.small, t2.medium, t2.large

Default: t2.small

Description: (Required) The EC2 instance type for the EC2Rescue instance. Recommended size: t2.small.

- AssumeRole

Type: String

Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

**Required IAM Permissions**

You must have at least `ssm:ExecuteAutomation, ssm:GetParameter` (to retrieve the SSH key parameter name) and `ssm:GetAutomationExecution` to be able to read the automation output. For more information about the required permissions, see [AWSSupport-StartEC2RescueWorkflow](#) (p. 512).

**Document Steps**

1. `aws:assertAwsResourceProperty` - Assert if the provided instance is Windows.
   a. (EC2Rescue for Windows) If the provided instance is Windows:
      i. `aws:executeAutomation` - Invoke AWSSupport-StartEC2RescueWorkflow with the EC2Rescue for Windows offline password reset script
      ii. `aws:executeAwsApi` - Retrieve the backup AMI ID from the nested automation
      iii. `aws:executeAwsApi` - Retrieve the password-enabled AMI ID from the nested automation
      iv. `aws:executeAwsApi` - Retrieve the EC2Rescue summary from the nested automation
   b. (EC2Rescue for Linux) If the provided instance is Linux:
      i. `aws:executeAutomation` - Invoke AWSSupport-StartEC2RescueWorkflow with the EC2Rescue for Linux offline SSH key injection script
      ii. `aws:executeAwsApi` - Retrieve the backup AMI ID from the nested automation
      iii. `aws:executeAwsApi` - Retrieve the SSM parameter name for the injected SSH key
      iv. `aws:executeAwsApi` - Retrieve the EC2Rescue summary from the nested automation

**Outputs**

- `getEC2RescueForWindowsResult.Output`
- `getWindowsBackupAmi.ImageId`
- `getWindowsPasswordEnabledAmi.ImageId`
- `getEC2RescueForLinuxResult.Output`
- `getLinuxBackupAmi.ImageId`
- `getLinuxSSHKeyParameter.Name`

_AWS-ReleaseElasticIP_

**Description**
Release the specified Elastic IP address using the allocation ID.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platform(s)
Windows, Linux

Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- AllocationId
  Type: String
  Description: (Required) The Allocation ID of the Elastic IP address.

AWS-ResizeInstance

Description
Change the instance type of an EC2 instance.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- InstanceId
  Type: String
  Description: The ID of the instance to resize.
Type: String
Description: (Required) The ID of the instance.

- **InstanceType**
  Type: String
  Description: (Required) The instance type.

- **LambdaAssumeRole**
  Type: String
  Description: (Optional) The ARN of the role assumed by Lambda.

---

**AWS-RestartEC2Instance**

**Description**
Restart one or more EC2 instances.

**Run this Automation (console)**

**Document Type**
Automation

**Owner**
Amazon

**Platforms**
Windows, Linux

**Parameters**

- **AutomationAssumeRole**
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- **InstanceId**
  Type: StringList
  Description: (Required) EC2 instance(s) to restart

---

**AWSSupport-SendLogBundleToS3Bucket**

**Description**
The AWSSupport-SendLogBundleToS3Bucket Automation document uploads a log bundle generated by the EC2Rescue tool from the target instance to the specified S3 bucket. The automation execution installs the platform specific version of EC2Rescue based on the platform of the target instance. EC2Rescue is then used to collect all the available operating system (OS) logs.

**Run this Automation (console)**
Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
- InstanceId
  Type: String
  Description: (Required) The ID of the Windows or Linux managed instance you want to collect logs from.
- S3BucketName
  Type: String
  Description: (Required) S3 bucket to upload the logs to.
- S3Path
  Type: String
  Default: AWSSupport-SendLogBundleToS3Bucket/
  Description: (Optional) S3 path for the collected logs.
- AutomationAssumeRole
  Type: String
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

Required IAM Permissions
It is recommended that the EC2 instance receiving the command has an IAM role with the AmazonSSMManagedInstanceCore Amazon managed policy attached. The user must have at least ssm:ExecuteAutomation and ssm:SendCommand to run the automation and send the command to the instance, plus ssm:GetAutomationExecution to be able to read the automation output.

Document Steps
1. aws:runCommand - Install EC2Rescue via AWS-ConfigureAWSPackage.
2. aws:runCommand - Run the PowerShell script to collect Windows troubleshooting logs with EC2Rescue.
3. aws:runCommand - Run the bash script to collect Linux troubleshooting logs with EC2Rescue.

Outputs
collectAndUploadWindowsLogBundle.Output
collectAndUploadLinuxLogBundle.Output
**AWS-SetupInventory**

**Description**

Create a Systems Manager Inventory association for one or more managed instances. The system collects metadata from your instances according to the schedule in the association. For more information, see *AWS Systems Manager Inventory (p. 704).*

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **Applications**
  
  Type: String
  
  Default: Enabled
  
  Description: (Optional) Collect metadata about installed applications.

- **AssociatedDocName**
  
  Type: String
  
  Default: AWS-GatherSoftwareInventory
  
  Description: (Optional) The name of the SSM document used to collect Inventory from the managed instance.

- **AssociationName**
  
  Type: String
  
  Description: (Optional) A name for the Inventory association that will be assigned to the instance.

- **AssocWaitTime**
  
  Type: String
  
  Default: PT5M
  
  Description: (Optional) Amount of time that Inventory collection should pause when the Inventory association start time is reached. The time uses ISO 8601 format.

- **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- **AwsComponents**
Type: String
Default: Enabled
Description: (Optional) Collect metadata for AWS Components like amazon-ssm-agent.
  - CustomInventory
    Type: String
    Default: Enabled
    Description: (Optional) Collect custom inventory metadata.
    - Files
      Type: String
      Description: (Optional) Collect metadata about files on your instances. For more information about how to collect this type of Inventory data, see Working with file and Windows registry inventory (p. 711). Requires SSMAgent version 2.2.64.0 or later. Linux example: [{"Path": "/usr/bin", "Pattern": ["aws", "ssm"], "Recursive": false}, {"Path": "/var/log", "Pattern": ["amazon*.*"], "Recursive": true, "DirScanLimit": 1000}] Windows example: [{"Path": "%PROGRAMFILES%", "Pattern": ["*.exe"], "Recursive": true}]
      - 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.
      - InstanceId
        Type: String
        Default: *
        Description: (Required) EC2 instances that you want to inventory.
    - LambdaAssumeRole
      Type: String
      Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.
    - NetworkConfig
      Type: String
      Default: Enabled
      Description: (Optional) Collect metadata about network configurations.
    - OutputS3BucketName
      Type: String
      Description: (Optional) Name of an Amazon S3 bucket where you want to write Inventory log data.
    - OutputS3KeyPrefix
      Type: String
Description: (Optional) An Amazon S3 key prefix (subfolder) where you want to write Inventory log data.

- **OutputS3Region**

  Type: String

  Description: (Optional) The name of the AWS Region where the Amazon S3 exists.

- **Schedule**

  Type: String

  Default: cron(0 */30 * * ? *)

  Description: (Optional) A cron expression for the Inventory association schedule. The default is every 30 minutes.

- **Services**

  Type: String

  Default: Enabled

  Description: (Optional, Windows OS only, requires SSMAgent version 2.2.64.0 and above) Collect data for service configurations.

- **WindowsRegistry**

  Type: String

  Description: (Optional) Collect metadata about Microsoft Windows Registry keys. For more information about how to collect this type of Inventory data, see Working with file and Windows registry inventory (p. 711). Requires SSMAgent version 2.2.64.0 or later. Example:

  ```json
  [{"Path":"HKEY_CURRENT_CONFIG\System","Recursive":true},{"Path":"HKEY_LOCAL_MACHINE\SOFTWARE\Amazon\MachineImage", "ValueNames":["AMIName"]}
  ```

- **WindowsRoles**

  Type: String

  Default: Enabled

  Description: (Optional) Collect information about Windows roles on the instance. Applies to Windows operating systems only. Requires SSMAgent version 2.2.64.0 or later.

- **WindowsUpdates**

  Type: String

  Default: Enabled

  Description: (Optional) Collect data about all Windows Updates on the instance.

---

**AWSSupport-SetupIPMonitoringFromVPC**

**Description**

`AWSSupport-SetupIPMonitoringFromVPC` creates an EC2 instance in the specified subnet and monitors selected target IPs (IPv4 or IPv6) by continuously running ping, MTR, traceroute and tracetcp tests. The results are stored in Amazon CloudWatch Logs logs, and metric filters are applied to quickly visualize latency and packet loss statistics in a CloudWatch dashboard.
Additional Information

The CloudWatch Logs data can be used for network troubleshooting and analysis of pattern/trends. Additionally, you can configure CloudWatch alarms with Amazon SNS notifications when packet loss and/or latency reach a threshold. The data can also be used when opening a Premium Support case, to help isolate an issue quickly and reduce time to resolution when investigating a network issue.

**Note**
To clean up resources created by AWSSupport-SetupIPMonitoringFromVPC, you can run the Automation document AWSSupport-TerminateIPMonitoringFromVPC. For more information, see AWSSupport-TerminateIPMonitoringFromVPC (p. 521).

Run this Automation (console)

**Document Type**
Automation

**Owner**
Amazon

**Parameters**

- **SubnetId**
  
  Type: String
  
  Description: (Required) The subnet ID for the monitor instance. Be aware that if you specify a private subnet, then you must make sure there is Internet access to allow the monitor instance to setup the test (meaning, install the CloudWatch Logs agent, interact with Systems Manager and CloudWatch).

- **TargetIPs**
  
  Type: String
  
  Description: (Required) Comma separated list of IPv4s and/or IPv6s to monitor. No spaces allowed. Maximum size is 255 characters. Be aware that if you provide an invalid IP, then the automation will fail and rollback the test setup.

- **CloudWatchLogGroupNamePrefix**
  
  Type: String
  
  Default: /AWSSupport-SetupIPMonitoringFromVPC
  
  Description: (Optional) Prefix used for each CloudWatch log group created for the test results.

- **CloudWatchLogGroupRetentionInDays**
  
  Type: String
  
  Allowed values: 1,3,5,7,14,30,60,90,120,150,180,365,400,545,731,1827,3653
  
  Default: 7
  
  Description: (Optional) Number of days you want to keep the network monitoring results for.

- **InstanceType**
  
  Type: String
  
  Allowed values: t2.micro, t2.small,t2.medium,t2.large
Default: t2.micro

Description: (Optional) The EC2 instance type for the EC2Rescue instance. Recommended size: t2.micro.

- AutomationAssumeRole

Type: String

Description: (Optional) The AWS Identity and Access Management (IAM) role for this execution. If no role is specified, then Systems Manager Automation uses the permissions of the user that runs this document.

Required IAM Permissions

It is recommended that the user who runs the automation have the AmazonSSMAutomationRole IAM managed policy attached. In addition, the user must have the following policy attached to their user account, group, or role:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:CreateRole",
        "iam:CreateInstanceProfile",
        "iam:GetRole",
        "iam:GetInstanceProfile",
        "iam:DetachRolePolicy",
        "iam:AttachRolePolicy",
        "iam:PassRole",
        "iam:AddRoleToInstanceProfile",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteInstanceProfile",
        "iam:PutRolePolicy",
        "iam:DeleteRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::AWS_account_ID:role/AWSSupport/SetupIPMonitoringFromVPC_*",
        "arn:aws:iam::AWS_account_ID:instance-profile/AWSSupport/SetupIPMonitoringFromVPC_*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:DetachRolePolicy",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::aws:policy/service-role/AmazonSSMManagedInstanceCore"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "cloudwatch:DeleteDashboards"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```
Document Steps

1. `aws:executeAwsApi` - describe the provided subnet.
2. `aws:branch` - evaluate the TargetIPs input.
   (IPv6) If TargetIPs contains an IPv6:
   - `aws:assertAwsResourceProperty` - check the provided subnet has an IPv6 pool associated
3. `aws:executeAwsApi` - get the latest Amazon Linux 2 AMI from Parameter Store.
4. `aws:executeAwsApi` - create a security group for the test in the subnet's VPC.
   (Cleanup) If the security group creation fails:
   - `aws:executeAwsApi` - delete the security group created by the automation, if it exists.
5. `aws:executeAwsApi` - allow all outbound traffic in the test security group.
   (Cleanup) If the security group egress rule creation fails:
   - `aws:executeAwsApi` - delete the security group created by the automation, if it exists.
6. `aws:executeAwsApi` - create an IAM role for the test EC2 instance
   (Cleanup) If the role creation fails:
   a. `aws:executeAwsApi` - delete the IAM role created by the automation, if it exists.
   b. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.
7. `aws:executeAwsApi` - attach the AmazonSSMManagedInstanceCore managed policy
   (Cleanup) If the policy attachment fails:
   a. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation, if attached.
   b. `aws:executeAwsApi` - delete the IAM role created by the automation.
   c. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.
8. `aws:executeAwsApi` - attach an inline policy to allow setting CloudWatch log group retentions and creating a CloudWatch dashboard
   (Cleanup) If the inline policy attachment fails:
   a. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation, if created.
   b. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   c. `aws:executeAwsApi` - delete the IAM role created by the automation.
d. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

9. `aws:executeAwsApi` - create an IAM instance profile.

   (Cleanup) If the instance profile creation fails:
   a. `aws:executeAwsApi` - delete the IAM instance profile created by the automation, if it exists.
   b. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
   c. `aws:executeAwsApi` - delete the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   d. `aws:executeAwsApi` - delete the IAM role created by the automation.
   e. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

10. `aws:executeAwsApi` - associate the IAM instance profile to the IAM role.

   (Cleanup) If the instance profile and role association fails:
   a. `aws:executeAwsApi` - remove the IAM instance profile from the role, if associated.
   b. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
   c. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
   d. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   e. `aws:executeAwsApi` - delete the IAM role created by the automation.
   f. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

11. `aws:sleep` - wait for the instance profile to become available.

12. `aws:runInstances` - create the test instance in the specified subnet, and with the instance profile created earlier attached.

   (Cleanup) If the step fails:
   a. `aws:changeInstanceState` - terminate the test instance.
   b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
   c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
   d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
   e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   f. `aws:executeAwsApi` - delete the IAM role created by the automation.
   g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

13. `aws:branch` - evaluate the TargetIPs input.

   (IPv6) If TargetIPs contains an IPv6:
   
   `aws:executeAwsApi` - assign an IPv6 to the test instance.


   (Cleanup) If the step fails:
   a. `aws:changeInstanceState` - terminate the test instance.
   b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
   c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
   d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
   e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   f. `aws:executeAwsApi` - delete the IAM role created by the automation.
   g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

15. `aws:runCommand` - install test pre-requisites:
(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.
b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
f. `aws:executeAwsApi` - delete the IAM role created by the automation.
g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

16 `aws:runCommand` - validate the provided IPs are syntactically correct IPv4 and/or IPv6 addresses:

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.
b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
f. `aws:executeAwsApi` - delete the IAM role created by the automation.
g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

17 `aws:runCommand` - define the MTR test for each of the provided IPs.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.
b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
f. `aws:executeAwsApi` - delete the IAM role created by the automation.
g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

18 `aws:runCommand` - define the first ping test for each of the provided IPs.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.
b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
f. `aws:executeAwsApi` - delete the IAM role created by the automation.
g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

19 `aws:runCommand` - define the second ping test for each of the provided IPs.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.
b. `aws:executeAwsApi` - remove the IAM instance profile from the role.
c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.

e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.

f. `aws:executeAwsApi` - delete the IAM role created by the automation.

g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

20 `aws:runCommand` - define the tracepath test for each of the provided IPs.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.

b. `aws:executeAwsApi` - remove the IAM instance profile from the role.

c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.

d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.

e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.

f. `aws:executeAwsApi` - delete the IAM role created by the automation.

g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

21 `aws:runCommand` - define the traceroute test for each of the provided IPs.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.

b. `aws:executeAwsApi` - remove the IAM instance profile from the role.

c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.

d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.

e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.

f. `aws:executeAwsApi` - delete the IAM role created by the automation.

g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

22 `aws:runCommand` - configure CloudWatch logs.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.

b. `aws:executeAwsApi` - remove the IAM instance profile from the role.

c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.

d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.

e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.

f. `aws:executeAwsApi` - delete the IAM role created by the automation.

g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.

23 `aws:runCommand` - schedule cronjobs to run each test every minute.

(Cleanup) If the step fails:

a. `aws:changeInstanceState` - terminate the test instance.

b. `aws:executeAwsApi` - remove the IAM instance profile from the role.

c. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.

d. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.

e. `aws:executeAwsApi` - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.

f. `aws:executeAwsApi` - delete the IAM role created by the automation.

g. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.
24. **aws:sleep** - wait for the tests to generate some data.
25. **aws:runCommand** - set the desired CloudWatch log group retentions.

(Cleanup) If the step fails:
   a. **aws:changeInstanceState** - terminate the test instance.
   b. **aws:executeAwsApi** - remove the IAM instance profile from the role.
   c. **aws:executeAwsApi** - delete the IAM instance profile created by the automation.
   d. **aws:executeAwsApi** - delete the CloudWatch inline policy from the role created by the automation.
   e. **aws:executeAwsApi** - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   f. **aws:executeAwsApi** - delete the IAM role created by the automation.
   g. **aws:executeAwsApi** - delete the security group created by the automation, if it exists.

26. **aws:runCommand** - set the CloudWatch log group metric filters.

(Cleanup) If the step fails:
   a. **aws:changeInstanceState** - terminate the test instance.
   b. **aws:executeAwsApi** - remove the IAM instance profile from the role.
   c. **aws:executeAwsApi** - delete the IAM instance profile created by the automation.
   d. **aws:executeAwsApi** - delete the CloudWatch inline policy from the role created by the automation.
   e. **aws:executeAwsApi** - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   f. **aws:executeAwsApi** - delete the IAM role created by the automation.
   g. **aws:executeAwsApi** - delete the security group created by the automation, if it exists.

27. **aws:runCommand** - create the CloudWatch dashboard.

(Cleanup) If the step fails:
   a. **aws:executeAwsApi** - delete the CloudWatch dashboard, if it exists.
   b. **aws:changeInstanceState** - terminate the test instance.
   c. **aws:executeAwsApi** - remove the IAM instance profile from the role.
   d. **aws:executeAwsApi** - delete the IAM instance profile created by the automation.
   e. **aws:executeAwsApi** - delete the CloudWatch inline policy from the role created by the automation.
   f. **aws:executeAwsApi** - detach the AmazonSSMManagedInstanceCore managed policy from the role created by the automation.
   g. **aws:executeAwsApi** - delete the IAM role created by the automation.
   h. **aws:executeAwsApi** - delete the security group created by the automation, if it exists.

**Outputs**

createCloudWatchDashboards.Output - the URL of the CloudWatch dashboard.
createManagedInstance.InstanceIds - the test instance ID.

**AWS-SetupManagedInstance**

**Description**

Configure an instance with an AWS Identity and Access Management (IAM) role for Systems Manager access.

**Run this Automation (console)**

**Document Type**
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- InstanceId
  Type: String
  Description: (Required) ID of the EC2 instance to configure
- LambdaAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.
- RoleName
  Type: String
  Default: SSMRoleForManagedInstance
  Description: (Optional) The name of the IAM role for the EC2 instance. If this role does not exist, it will be created. When specifying this value, verify that the role contains the AmazonSSMManagedInstanceCore Managed Policy.

AWS-SetupManagedRoleOnEC2Instance

Description
Configure an instance with the SSMRoleForManagedInstance managed IAM role for Systems Manager access.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux
Parameters

- **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- **InstanceId**
  
  Type: String
  
  Description: (Required) ID of the EC2 instance to configure

- **LambdaAssumeRole**
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Lambda created by Automation to perform the actions on your behalf. If not specified a transient role will be created to run the Lambda function.

- **RoleName**
  
  Type: String
  
  Default: SSMRoleForManagedInstance
  
  Description: (Optional) The name of the IAM role for the EC2 instance. If this role does not exist, it will be created. When specifying this value, verify that the role contains the AmazonSSMManagedInstanceCore Managed Policy.

**AWSEC2-SQLServerDBRestore**

Description

The AWSEC2-SQLServerDBRestore document restores Microsoft SQL Server database backups stored in Amazon S3 to SQL Server 2017 running on an Amazon Elastic Compute Cloud (EC2) Linux instance. You may provide your own EC2 instance running SQL Server 2017 Linux. If an EC2 instance is not provided, the automation workflow launches and configures a new Ubuntu 16.04 EC2 instance with SQL Server 2017. The automation supports restoring full, differential, and transactional log backups. This automation accepts multiple database backup files and automatically restores the most recent valid backup of each database in the files provided.

To automate both backup and restore of an on-premises SQL Server database to an EC2 instance running SQL Server 2017 Linux, you can use the AWS-signed PowerShell script MigrateSQLServerToEC2Linux.

**Important**

This automation workflow resets the SQL Server server administrator (SA) user password every time the workflow runs. After the automation workflow is complete, you must set your own SA user password again before you connect to the SQL Server instance.

Run this Automation (console)

Document Type

Automation

Owner

Amazon
Platforms

Linux

Prerequisites

- This Automation document only works with EC2 instances for Linux running SQL Server.
- This Automation workflow must be run by a user with, at minimum, the permissions designated in the Required IAM Permissions section below.
- If you are providing your own EC2 instance:
  - Configure the EC2 instance with an AWS Identity and Access Management (IAM) instance profile that has the AmazonSSMManagedInstanceCore managed policy attached. For more information, see Create an IAM instance profile for Systems Manager (p. 30).
  - Verify that SSM Agent is installed on your EC2 instance. For more information, see Installing and configuring SSM Agent on EC2 instances for Linux (p. 69).
  - Verify that the EC2 instance has enough free disk space to download and restore the SQL Server backups.

Limitations

This automation does not support restoring to SQL Server running on EC2 instances for Windows Server. This automation only restores database backups that are compatible with SQL Server Linux 2017. For more information, see Editions and Supported Features of SQL Server 2017 on Linux.

Required IAM Permissions

The user who runs the Automation workflow must have the following permissions:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ec2:DescribeImages",
            "ec2:RunInstances",
            "ec2:CreateTags",
            "ec2:DescribeInstances",
            "ec2:DescribeInstanceStatus",
            "ec2:RebootInstances",
            "ssm:SendCommand",
            "ssm:GetAutomationExecution",
            "ssm:ListCommands",
            "ssm:StartAutomationExecution",
            "ssm:DescribeInstanceInformation",
            "ssm:ListCommandInvocations",
            "iam:PassRole"
         ],
         "Resource": "*"
      }
   }
}
```

Parameters

- **S3Input**
  
  Type: String
Description: (Required) S3 bucket name, comma-separated list of S3 object keys, or comma-separated list of pre-signed S3 URLs containing the SQL backup files to be restored.

- IsS3PresignedUrl
  
  Type: String
  
  Description: (Optional) If S3Input is a pre-signed S3 URL, indicate “yes”.
  
  Default value: "no"
  
  Allowed values: "yes", "no"

- InstanceId
  
  Type: String
  
  Description: (Optional) The instance running SQL Server 2017 on Linux. If no InstanceId is provided, the automation launches a new EC2 instance using the InstanceType and SQLServerEdition provided.

- InstanceType
  
  Type: String
  
  Description: (Optional) The instance type of the EC2 instance to be launched.

- SQLServerEdition
  
  Type: String
  
  Description: (Optional) The edition of SQL Server 2017 to be installed on the newly created EC2 instance.
  
  Allowed values: "Standard", "Enterprise", "Web", "Express"

- SubnetId
  
  Type: String
  
  Description: (Optional) The subnet in which to launch the new EC2 instance. The subnet must have outbound connectivity to AWS services. If a value for SubnetId is not provided, the automation uses the default subnet.

- IamInstanceProfileName
  
  Type: String
  
  Description: (Optional) The IAM instance profile to attach to the new EC2 instance. The IAM instance profile must have the AmazonSSMManagedInstanceCore managed policy attached.

- DataDirectorySize
  
  Type: String
  
  Description: (Optional) Desired volume size (GiB) of the SQL Server Data directory for the new EC2 instance.
  
  Default value: 100

- LogDirectorySize
  
  Type: String
  
  Description: (Optional) Desired volume size (GiB) of the SQL Server Log directory for the new EC2 instance.
Default value: 100

- **TempDbDirectorySize**
  
  Type: String
  
  Description: (Optional) Desired volume size (GiB) of the SQL Server TempDB directory for the new EC2 instance.
  
  Default value: 100

- **DatabaseNames**
  
  Type: String
  
  Description: (Optional) Comma-separated list of the names of databases to restore.

- **KeyPair**
  
  Type: String
  
  Description: (Optional) Key pair to use when creating the new EC2 instance.

**Document Steps**

**For new EC2 instances:**

1. `aws:executeAwsApi` - Retrieve the AMI ID for SQL Server 2017 on Ubuntu 16.04.
2. `aws:runInstances` - Launch a new EC2 instance for Linux.
3. `aws:waitForAwsResourceProperty` - Wait for the newly created EC2 instance to be ready.
4. `aws:executeAwsApi` - Reboot the instance if the instance is not ready.
5. `aws:assertAwsResourceProperty` - Verify that SSM Agent is installed.
6. `aws:runCommand` - Run the SQL Server restore script in PowerShell.

**For existing EC2 instances:**

1. `aws:waitForAwsResourceProperty` - Verify that the EC2 instance is ready.
2. `aws:executeAwsApi` - Reboot the instance if the instance is not ready.
3. `aws:assertAwsResourceProperty` - Verify that SSM Agent is installed.
4. `aws:runCommand` - Run the SQL Server restore script in PowerShell.

**Outputs**

- `getInstance.InstanceId`
- `restoreToNewInstance.Output`
- `restoreToExistingInstance.Output`

**AWS—StartEC2Instance**

**Description**

Start one or more EC2 instances.

**Run this Automation (console)**
**Document Type**
Automation

**Owner**
Amazon

**Platforms**
Windows, Linux

**Parameters**
- **AutomationAssumeRole**
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- **InstanceId**
  Type: StringList
  Description: (Required) EC2 instances to start.

**Outputs**
The automation execution has no output.

**AWSSupport-StartEC2RescueWorkflow**

**Description**
The AWSSupport-StartEC2RescueWorkflow automation document runs the provided base64 encoded script (Bash or Powershell) on a helper instance created to rescue your instance. The root volume of your instance is attached and mounted to the helper instance, also known as the EC2Rescue instance. If your instance is Windows, provide a Powershell script. Otherwise, use Bash. The workflow sets some environment variables which you can use in your script. The environment variables contain information about the input you provided, as well as information about the offline root volume. The offline volume is already mounted and ready to use. For example, you can save a Desired State Configuration file to an offline Windows root volume, or chroot to an offline Linux root volume and perform an offline remediation.

**Run this Automation (console)**

**Important**
Amazon EC2 instances created from Marketplace Amazon Machine Images (AMIs) are not supported by this automation.

**Additional Information**

To base64 encode a script, you can use either Powershell or Bash. Powershell:

```powershell
[System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes([System.IO.File]::ReadAllText('PATH_TO_FILE')))
```

Bash:

```bash
512
```
Here is a list of environment variables you can use in your offline scripts, depending on the target OS:

### Windows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$env:EC2RESCUE_ACCOUNT_ID</td>
<td>${{ global:ACCOUNT_ID }}</td>
<td>123456789012</td>
</tr>
<tr>
<td>$env:EC2RESCUE_DATE</td>
<td>${{ global:DATE }}</td>
<td>123456789012</td>
</tr>
<tr>
<td>$env:EC2RESCUE_DATE_TIME</td>
<td>${{ global:DATE_TIME }}</td>
<td>123456789012_18.09.59</td>
</tr>
<tr>
<td>$env:EC2RESCUE_EC2RW_DIR</td>
<td>EC2Rescue for Windows installation path</td>
<td>C:\Program Files\Amazon \EC2Rescue</td>
</tr>
<tr>
<td>$env:EC2RESCUE_EXECUTION_ID</td>
<td>${{ automation:EXECUTION_ID }}</td>
<td>7ef8008e-219b-4aca-8bb5-65e2e898e20b</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_CURRENT_CONTROL_SET</td>
<td>Offline Windows Current Control Set path</td>
<td>HKLM:\AWSTempSystem \ControlSet001</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_DRIVE</td>
<td>Offline Windows drive letter</td>
<td>D:\</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_EBS_DEVICE</td>
<td>Offline root volume EBS device</td>
<td>xvdf</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_KERNEL_VER</td>
<td>Offline Windows Kernel version</td>
<td>6.1.7601.24214</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_OS_ARCHITECTURE</td>
<td>Offline Windows architecture</td>
<td>AMD64</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_OS_CAPTION</td>
<td>Offline Windows caption</td>
<td>Windows Server 2008 R2 Datacenter</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_OS_TYPE</td>
<td>Offline Windows OS type</td>
<td>Server</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_PROGRAM_FILES_DIR</td>
<td>Offline Program files directory path</td>
<td>D:\Program Files</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_PROGRAM_FILES_X86_DIR</td>
<td>Offline Program files x86 directory path</td>
<td>D:\Program Files (x86)</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_REGISTRY_DIR</td>
<td>Offline Windows registry directory path</td>
<td>D:\Windows\System32\config</td>
</tr>
<tr>
<td>$env:EC2RESCUE_OFFLINE_SYSTEM_ROOT</td>
<td>Offline Windows system root directory path</td>
<td>D:\Windows</td>
</tr>
<tr>
<td>$env:EC2RESCUE_REGION</td>
<td>${{ global:REGION }}</td>
<td>us-west-1</td>
</tr>
<tr>
<td>$env:EC2RESCUE_S3_BUCKET</td>
<td>${{ S3BucketName }}</td>
<td>mybucket</td>
</tr>
<tr>
<td>$env:EC2RESCUE_S3_PREFIX</td>
<td>${{ S3Prefix }}</td>
<td>myprefix/</td>
</tr>
<tr>
<td>$env:EC2RESCUE_SOURCE_INSTANCE</td>
<td>${{ InstanceId }}</td>
<td>i-abcdefgh123456789012</td>
</tr>
<tr>
<td>$script:EC2RESCUE_OFFLINE_WINDOWS_INSTALL_METADATA</td>
<td>Customer Powershell Object</td>
<td></td>
</tr>
</tbody>
</table>
Linux:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Example value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2RESCUE_ACCOUNT_ID</td>
<td>{{ global:ACCOUNT_ID }}</td>
<td>123456789012</td>
</tr>
<tr>
<td>EC2RESCUE_DATE</td>
<td>{{ global:DATE }}</td>
<td>2018-09-07</td>
</tr>
<tr>
<td>EC2RESCUE_DATE_TIME</td>
<td>{{ global:DATE_TIME }}</td>
<td>2018-09-07_18.09.59</td>
</tr>
<tr>
<td>EC2RESCUE_EC2RL_DIR</td>
<td>EC2Rescue for Linux installation path</td>
<td>/usr/local/ec2rl-1.1.3</td>
</tr>
<tr>
<td>EC2RESCUE_EXECUTION_ID</td>
<td>{{ automation:EXECUTION_ID }}</td>
<td>7ef8008e-219b-4aca-8bb5-65e2e898e20b</td>
</tr>
<tr>
<td>EC2RESCUE_OFFLINE_DEVICE</td>
<td>Offline device name</td>
<td>/dev/xvdf1</td>
</tr>
<tr>
<td>EC2RESCUE_OFFLINE_EBS_DEVICE</td>
<td>Offline root volume EBS device</td>
<td>/dev/sdf</td>
</tr>
<tr>
<td>EC2RESCUE_OFFLINE_SYSTEM_ROOT</td>
<td>Offline root volume mount point</td>
<td>/mnt/mount</td>
</tr>
<tr>
<td>EC2RESCUE_PYTHON</td>
<td>Python version</td>
<td>python2.7</td>
</tr>
<tr>
<td>EC2RESCUE_REGION</td>
<td>{{ global:REGION }}</td>
<td>us-west-1</td>
</tr>
<tr>
<td>EC2RESCUE_S3_BUCKET</td>
<td>{{ S3BucketName }}</td>
<td>mybucket</td>
</tr>
<tr>
<td>EC2RESCUE_S3_PREFIX</td>
<td>{{ S3Prefix }}</td>
<td>myprefix/</td>
</tr>
<tr>
<td>EC2RESCUE_SOURCE_INSTANCE</td>
<td>{{ InstanceId }}</td>
<td>i-abcddefgh123456789</td>
</tr>
</tbody>
</table>

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- **InstanceId**
  
  **Type:** String
  
  **Description:** (Required) ID of your EC2 instance. IMPORTANT: AWS Systems Manager Automation stops this instance. Data stored in instance store volumes will be lost. The public IP address will change if you are not using an Elastic IP.

- **OfflineScript**
  
  **Type:** String
  
  **Description:** (Required) Base64 encoded script to run against the helper instance. Use Bash if your source instance is Linux, and PowerShell if it is Windows.

- **EC2RescueInstanceType**
  
  **Type:** String
Allowed values: t2.small,t2.medium,t2.large

Default: t2.small

Description: (Optional) The EC2 instance type for the EC2Rescue instance.

- SubnetId
  Type: String
  Default: SelectedInstanceSubnet

Description: (Optional) The subnet ID for the EC2Rescue instance. By default, the same subnet where the provided instance resides is used. IMPORTANT: If you provide a custom subnet, it must be in the same Availability Zone as InstanceId, and it must allow access to the SSM endpoints.

- S3BucketName
  Type: String

Description: (Optional) S3 bucket name in your account where you want to upload the troubleshooting logs. Make sure the bucket policy does not grant unnecessary read/write permissions to parties that do not need access to the collected logs.

- S3Prefix
  Type: String
  Default: AWSSupport-EC2Rescue

Description: (Optional) A prefix for the S3 logs.

- AMIPrefix
  Type: String
  Default: AWSSupport-EC2Rescue

Description: (Optional) A prefix for the backup AMI name.

- CreatePreEC2RescueBackup
  Type: String
  Allowed values: True,False
  Default: False

Description: (Optional) Set it to True to create an AMI of InstanceId before running the script. The AMI will persist after the automation completes. It is your responsibility to secure access to the AMI, or to delete it.

- CreatePostEC2RescueBackup
  Type: String
  Allowed values: True,False
  Default: False

Description: (Optional) Set it to True to create an AMI of InstanceId after running the script, before starting it. The AMI will persist after the automation completes. It is your responsibility to secure access to the AMI, or to delete it.

- UniqueId
Type: String

Default: {{ automation:EXECUTION_ID }}

Description: (Optional) A unique identifier for the workflow.

- AutomationAssumeRole
  
  Type: String

Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to run this document.

Required IAM Permissions

It is recommended the user who runs the automation have the AmazonSSMAutomationRole IAM managed policy attached. In addition to that policy, the user must have:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction"
            ],
            "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-*",
            ]
        }
    ]
}
```
Document Steps

1. aws:executeAwsApi - Describe the provided instance
2. aws:executeAwsApi - Describe the provided instance's root volume
3. aws:assertAwsResourceProperty - Check the root volume device type is EBS
4. aws:assertAwsResourceProperty - Check the root volume is not encrypted
5. aws:assertAwsResourceProperty - Check the provide subnet ID
   a. (Use current instance subnet) - If *SubnetId = SelectedInstanceSubnet* then run aws:createStack to deploy the EC2Rescue CloudFormation stack
   b. (Create new VPC) - If *SubnetId = CreateNewVPC* then run aws:createStack to deploy the EC2Rescue CloudFormation stack
   c. (Use custom subnet) - In all other cases:
      - aws:assertAwsResourceProperty - Check the provided subnet is in the same Availability Zone as the provided instance
      - aws:createStack - Deploy the EC2Rescue CloudFormation stack
6. aws:invokeLambdaFunction - Perform additional input validation
7. aws:executeAwsApi - Update the EC2Rescue CloudFormation stack to create the EC2Rescue helper instance
8. aws:waitForAwsResourceProperty - Wait for the EC2Rescue CloudFormation stack update to complete
9. aws:executeAwsApi - Describe the EC2Rescue CloudFormation stack output to obtain the EC2Rescue helper instance ID
10. aws:waitForAwsResourceProperty - Wait for the EC2Rescue helper instance to become a managed instance
11. aws:changeInstanceState - Stop the provided instance
12aws:changeInstanceState - Stop the provided instance
13aws:changeInstanceState - Force stop the provided instance
14aws:assertAwsResourceProperty - Check the CreatePreEC2RescueBackup input value
   a. (Create pre-EC2Rescue backup) - If *CreatePreEC2RescueBackup = True*
   b. aws:executeAwsApi - Create an AMI backup of the provided instance
   c. aws:createTags - Tag the AMI backup
15aws:runCommand - Install EC2Rescue on the EC2Rescue helper instance
16aws:executeAwsApi - Detach the root volume from the provided instance
17aws:assertAwsResourceProperty - Check the provided instance platform
   a. (Instance is Windows):
      - aws:executeAwsApi - Attach the root volume to the EC2Rescue helper instance as *xvdf*
      - aws:sleep - Sleep 10 seconds
      - aws:runCommand - Run the provided offline script in Powershell
   b. (Instance is Linux):
      - aws:executeAwsApi - Attach the root volume to the EC2Rescue helper instance as */dev/sdf*
      - aws:sleep - Sleep 10 seconds
      - aws:runCommand - Run the provided offline script in Bash
18aws:changeInstanceState - Stop the EC2Rescue helper instance
19aws:changeInstanceState - Force stop the EC2Rescue helper instance
20aws:executeAwsApi - Detach the root volume from the EC2Rescue helper instance
21aws:executeAwsApi - Attach the root volume back to the provided instance
22aws:assertAwsResourceProperty - Check the CreatePostEC2RescueBackup input value
   a. (Create post-EC2Rescue backup) - If *CreatePostEC2RescueBackup = True*
   b. aws:executeAwsApi - Create an AMI backup of the provided instance
   c. aws:createTags - Tag the AMI backup
23aws:executeAwsApi - Restore the initial delete on termination state for the root volume of the provided instance
24aws:changeInstanceState - Restore the initial state of the provided instance (running/stopped)
25aws:deleteStack - Delete the EC2Rescue CloudFormation stack

**Outputs**

runScriptForLinux.Output
runScriptForWindows.Output
preScriptBackup.ImageId
postScriptBackup.ImageId

**AWS-StartRDSInstance**

**Description**

Start an Amazon Relational Database Service (Amazon RDS) instance.
Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

• AutomationAssumeRole
  
  Type: String

  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

• InstanceId
  
  Type: String

  Description: (Required) ID of the Amazon RDS instance to start.

AWS-StopEC2Instance

Description

Stop one or more EC2 instances.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

• AutomationAssumeRole
  
  Type: String

  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

• InstanceId
  
  Type: StringList
Description: (Required) IDs of one or more EC2 instances to stop

AWS-StopRDSInstance

Description
Stop an Amazon Relational Database Service (Amazon RDS) instance. This document calls the StopDBInstance API action. The StopDBInstance API doesn't apply to Aurora MySQL and Aurora PostgreSQL. For Aurora clusters, you must use the StopDBCluster API action instead.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
- AutomationAssumeRole
  Type: String
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.
- InstanceId
  Type: String
  Description: (Required) ID of the Amazon RDS instance to stop.

AWS-TerminateEC2Instance

Description
Terminate one or more EC2 instances.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux
Parameters

- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf.

- InstanceId
  
  Type: StringList
  
  Description: (Required) IDs of one or more EC2 instances to terminate.

AWSSupport-TerminateIPMonitoringFromVPC

Description

AWSSupport-TerminateIPMonitoringFromVPC terminates an IP monitoring test previously started by AWSSupport-SetupIPMonitoringFromVPC. Data related to the specified test ID will be deleted.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Parameters

- AutomationExecutionId
  
  Type: String
  
  Description: (Required) AWSSupport-SetupIPMonitoringFromVPC automation execution ID of the test you want to terminate.

- SubnetId
  
  Type: String
  
  Description: (Required) The subnet ID for the monitor instance.

- InstanceId
  
  Type: String
  
  Description: (Required) The instance ID for the monitor instance.

- AutomationAssumeRole
  
  Type: String
  
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

Required IAM Permissions
It is recommended that the user who runs the automation have the `AmazonSSMAutomationRole` IAM managed policy attached. In addition, the user must have the following policy attached to their user account, group, or role:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:DetachRolePolicy",
        "iam:RemoveRoleFromInstanceProfile",
        "iam:DeleteRole",
        "iam:DeleteInstanceProfile",
        "iam:DeleteRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::An-AWS-Account-ID:role/AWSSupport/SetupIPMonitoringFromVPC_*",
        "arn:aws:iam::An-AWS-Account-ID:instance-profile/AWSSupport/SetupIPMonitoringFromVPC_*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "iam:DetachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::aws:policy/service-role/AmazonSSMManagedInstanceCore"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "cloudwatch:DeleteDashboards"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

**Document Steps**

1. `aws:assertAwsResourceProperty` - check AutomationExecutionId and InstanceId are related to the same test.
2. `aws:assertAwsResourceProperty` - check SubnetId and InstanceId are related to the same test.
3. `aws:executeAwsApi` - retrieve the test security group.
4. `aws:executeAwsApi` - delete the CloudWatch dashboard.
5. `aws:changeInstanceState` - terminate the test instance.
6. `aws:executeAwsApi` - remove the IAM instance profile from the role.
7. `aws:executeAwsApi` - delete the IAM instance profile created by the automation.
8. `aws:executeAwsApi` - delete the CloudWatch inline policy from the role created by the automation.
9. `aws:executeAwsApi` - detach the `AmazonSSMManagedInstanceCore` managed policy from the role created by the automation.
10. `aws:executeAwsApi` - delete the IAM role created by the automation.
11. `aws:executeAwsApi` - delete the security group created by the automation, if it exists.
Outputs

None

AWSSupport-TroubleshootConnectivityToRDS

Description

The AWSSupport-TroubleshootConnectivityToRDS Automation document diagnoses connectivity issues between an EC2 instance and an Amazon Relational Database Service instance. The automation ensures the DB instance is available, and then checks the associated security group rules, network access control lists (network ACLs), and route tables for potential connectivity issues.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Parameters

- AutomationAssumeRole
  Type: String
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.
- DBInstanceId
  Type: String
  Description: (Required) The DB instance ID to test connectivity to.
- SourceInstance
  Type: String
  Allowed pattern: ^i-[a-z0-9]{8,17}$
  Description: (Required) The ID of the EC2 instance to test connectivity from.

Required IAM Permissions

The AutomationAssumeRole requires the following actions to successfully execute the Automation document.

- ec2:DescribeInstances
- ec2:DescribeNetworkAcls
- ec2:DescribeRouteTables
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets
- rds:DescribeDBInstances

Document Steps
• aws:assertAwsResourceProperty - Confirms the DB instance status is available.
• aws:executeAwsApi - Gets information about the DB instance.
• aws:executeAwsApi - Gets information about the DB instance network ACLs.
• aws:executeAwsApi - Gets the DB instance subnet CIDR.
• aws:executeAwsApi - Gets information about the EC2 instance.
• aws:executeAwsApi - Gets information about the EC2 instance network ACLs.
• aws:executeAwsApi - Gets information about the security groups associated with the EC2 instance.
• aws:executeAwsApi - Gets information about the security groups associated with the DB instance.
• aws:executeAwsApi - Gets information about the route tables associated with the EC2 instance.
• aws:executeAwsApi - Gets information about the main route table associated with the Amazon VPC for the EC2 instance.
• aws:executeAwsApi - Gets information about the route tables associated with the DB instance.
• aws:executeAwsApi - Gets information about the main route table associated with the Amazon VPC for the DB instance.
• aws:executeScript - Evaluates security group rules.
• aws:executeScript - Evaluates network ACLs.
• aws:executeScript - Evaluates route tables.
• aws:sleep - Ends the automation execution.

**Outputs**

getRDSInstanceProperties.DBInstanceIdentifier - The DB instance used in the automation execution.

getRDSInstanceProperties.DBInstanceStatus - The current status of the DBInstance.


evalNetworkAclRules.NetworkAclEvaluation - Results from comparing the SourceInstance network ACLs to the DB instance network ACLs.

evalRouteTableEntries.RouteTableEvaluation - Results from comparing the SourceInstance route table to the DB instance routes.

**AWSSupport-TroubleshootDirectoryTrust**

**Description**

The AWSSupport-TroubleshootDirectoryTrust Automation document diagnoses trust creation issues between an AWS Managed Microsoft AD and a Microsoft Active Directory. The automation ensures the directory type supports trusts, and then checks the associated security group rules, network access control lists (network ACLs), and route tables for potential connectivity issues.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon
Parameters

- **AutomationAssumeRole**
  
  Type: String
  
  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

- **DirectoryId**
  
  Type: String
  
  Allowed pattern: ^d-[a-z0-9]{10}$
  
  Description: (Required) The ID of the AWS Managed Microsoft AD to troubleshoot.

- **RemoteDomainCidrs**
  
  Type: StringList
  
  Allowed values: \^([0-9]{1-9})\([0-9]{1-9}\)\([0-9]{1-9}\)\(2\)\(2[0-4][0-9]\)\(0-9\)\(25[0-5]\)\(3\)\([0-9]{1-9}\)\([0-9]{1-9}\)\([0-9]{1-9}\)\(2\)\(2[0-4][0-9]\)\(25[0-5]\)\(3\)\([0-9]{1-9}\)$
  
  Description: (Required) The CIDR(s) of the remote domain you are attempting to establish a trust relationship with. You can add multiple CIDRs using comma-separated values. For example, 172.31.48.0/20, 192.168.1.10/32.

- **RemoteDomainName**
  
  Type: String
  
  Description: (Required) The fully qualified domain name of the remote domain you are establishing a trust relationship with.

- **RequiredTrafficACL**
  
  Type: String
  
  Description: (Required) The default port requirements for AWS Managed Microsoft AD. In most cases, you should not modify the default value.
  

- **RequiredTrafficSG**
  
  Type: String
  
  Description: (Required) The default port requirements for AWS Managed Microsoft AD. In most cases, you should not modify the default value.
  

- **TrustId**
  
  Type: String
  
  Description: (Optional) The ID of the trust relationship to troubleshoot.

Required IAM Permissions
The AutomationAssumeRole requires the following actions to successfully execute the Automation document.

- ds:DescribeConditionalForwarders
- ds:DescribeDirectories
- ds:DescribeTrusts
- ds:ListIpRoutes
- ec2:DescribeNetworkAcls
- ec2:DescribeSecurityGroups
- ec2:DescribeSubnets

**Document Steps**

- aws:assertAwsResourceProperty - Confirms the directory type is AWS Managed Microsoft AD.
- aws:executeAwsApi - Gets information about the AWS Managed Microsoft AD.
- aws:branch - Branches workflow if a value is provided for the TrustId input parameter.
- aws:executeAwsApi - Gets information about the trust relationship.
- aws:executeAwsApi - Gets the conditional forwarder DNS IP addresses for the RemoteDomainName.
- aws:executeAwsApi - Gets information about IP routes that have been added to the AWS Managed Microsoft AD.
- aws:executeAwsApi - Gets the CIDRs of the AWS Managed Microsoft AD subnets.
- aws:executeAwsApi - Gets information about the security groups associated with the AWS Managed Microsoft AD.
- aws:executeAwsApi - Gets information about the network ACLs associated with the AWS Managed Microsoft AD.
- aws:executeScript - Confirms the RemoteDomainCidrs are valid values. Confirms that the AWS Managed Microsoft AD has conditional forwarders for the RemoteDomainCidrs, and that the requisite IP routes have been added to the AWS Managed Microsoft AD if the RemoteDomainCidrs are non-RFC 1918 IP addresses.
- aws:executeScript - Evaluates security group rules.
- aws:executeScript - Evaluates network ACLs.

**Outputs**

- evalDirectorySecurityGroup.output - Results from evaluating whether the security group rules associated with the AWS Managed Microsoft AD allow the requisite traffic for trust creation.
- evalAclEntries.output - Results from evaluating whether the network ACLs associated with the AWS Managed Microsoft AD allow the requisite traffic for trust creation.
- evaluateRemoteDomainCidr.output - Results from evaluating whether the RemoteDomainCidrs are valid values. Confirms that the AWS Managed Microsoft AD has conditional forwarders for the RemoteDomainCidrs, and that the requisite IP routes have been added to the AWS Managed Microsoft AD if the RemoteDomainCidrs are non-RFC 1918 IP addresses.

**AWSSupport-TroubleshootRDP**

**Description**

The AWSSupport-TroubleshootRDP automation document allows the user to check or modify common settings on the target instance which may impact Remote Desktop Protocol (RDP) connections, such as
the RDP port, Network Layer Authentication (NLA) and Windows Firewall profiles. Optionally, changes can be applied offline by stopping and starting the instance, if the user explicitly allows for offline remediation. By default, the document reads and outputs the values of the settings.

**Important**
Changes to the RDP settings, RDP service and Windows Firewall profiles should be carefully reviewed before running this document.

**Run this Automation (console)**

**Document Type**
Automation

**Owner**
Amazon

**Platforms**
Windows

**Parameters**

- **InstanceId**
  
  Type: String
  
  Description: (Required) The ID of the instance to troubleshoot the RDP settings of.

- **Action**
  
  Type: String
  
  Allowed values: CheckAll, FixAll, Custom
  
  Default: Custom
  
  Description: (Optional) [Custom] Use the values from Firewall, RDPServiceStartupType, RDPServiceAction, RDPPortAction, NLASettingAction and RemoteConnections to manage the settings. [CheckAll] Read the values of the settings without changing them. [FixAll] Restore RDP default settings, and disable the Windows Firewall.

- **AllowOffline**
  
  Type: String
  
  Allowed values: True, False
  
  Default: False
  
  Description: (Optional) Fix only - Set it to true if you allow an offline RDP remediation in case the online troubleshooting fails, or the provided instance is not a managed instance. Note: For the offline remediation, SSM Automation stops the instance, and creates an AMI before attempting any operations.

- **Firewall**
  
  Type: String
  
  Allowed values: Check, Disable
  
  Default: Check
Description: (Optional) Check or disable the Windows firewall (all profiles).

- **RDPServiceAction**
  
  Type: String
  
  Allowed values: Check, Start, Restart, Force-Restart
  
  Default: Check
  
  Description: (Optional) Check, start, restart, or force-restart the RDP service (TermService).

- **RDPServiceStartupType**
  
  Type: String
  
  Allowed values: Check, Auto
  
  Default: Check
  
  Description: (Optional) Check or set the RDP service to automatically start when Windows boots.

- **RDPPortAction**
  
  Type: String
  
  Allowed values: Check, Modify
  
  Default: Check
  
  Description: (Optional) Check the current port used for RDP connections, or modify the RDP port back to 3389 and restart the service.

- **NLASettingAction**
  
  Type: String
  
  Allowed values: Check, Disable
  
  Default: Check
  
  Description: (Optional) Check or disable Network Layer Authentication (NLA).

- **RemoteConnections**
  
  Type: String
  
  Allowed values: Check, Enable
  
  Default: Check
  
  Description: (Optional) An action to perform on the fDenyTSTCConnections setting: Check, Enable.

- **SubnetId**
  
  Type: String
  
  Default: SelectedInstanceSubnet
  
  Description: (Optional) Offline only - The subnet ID for the EC2Rescue instance used to perform the offline troubleshooting. If no subnet ID is specified, AWS Systems Manager Automation will create a new VPC. IMPORTANT: The subnet must be in the same Availability Zone as InstanceId, and it must allow access to the SSM endpoints.

- **S3BucketName**
Type: String

Description: (Optional) Offline only - S3 bucket name in your account where you want to upload the troubleshooting logs. Make sure the bucket policy does not grant unnecessary read/write permissions to parties that do not need access to the collected logs.

- AutomationAssumeRole
  
  Type: String

  Description: (Optional) The IAM role for this execution. If no role is specified, AWS Systems Manager Automation will use the permissions of the user that runs this document.

Required IAM Permissions

It is recommended that the EC2 instance receiving the command has an IAM role with the AmazonSSMManagedInstanceCore Amazon managed policy attached. For the online remediation, the user must have at least ssm:DescribeInstanceInformation, ssm:ExecuteAutomation and ssm:SendCommand to run the automation and send the command to the instance, plus ssm:GetAutomationExecution to be able to read the automation output. For the offline remediation, the user must have at least ssm:DescribeInstanceInformation, ssm:ExecuteAutomation, ec2:DescribeInstances, plus ssm:GetAutomationExecution to be able to read the automation output. AWSSupport-TroubleshootRDP calls AWSSupport-ExecuteEC2Rescue to perform the offline remediation - please review the permissions for AWSSupport-ExecuteEC2Rescue to ensure you can run the automation successfully.

Document Steps

1. aws:assertAwsResourceProperty - Check if the instance is a Windows Server instance
2. aws:assertAwsResourceProperty - Check if the instance is a managed instance
3. (Online troubleshooting) If the instance is a managed instance, then:
   a. aws:assertAwsResourceProperty - Check the provided Action value
   b. (Online check) If the Action = CheckAll, then:
      aws:runPowerShellScript - Runs the PowerShell script to get the Windows Firewall profiles status.
      aws:executeAutomation - Calls AWSSupport-ManageWindowsService to get the RDP service status.
      aws:executeAutomation - Calls AWSSupport-ManageRDPSettings to get the RDP settings.
   c. (Online fix) If the Action = FixAll, then:
      aws:runPowerShellScript - Runs the PowerShell script to disable all Windows Firewall profiles.
      aws:executeAutomation - Calls AWSSupport-ManageWindowsService to start the RDP service.
      aws:executeAutomation - Calls AWSSupport-ManageRDPSettings to enable remote connections and disable NLA.
   d. (Online management) If the Action = Custom, then:
      aws:runPowerShellScript - Runs the PowerShell script to manage the Windows Firewall profiles.
      aws:executeAutomation - Calls AWSSupport-ManageWindowsService to manage the RDP service.
      aws:executeAutomation - Calls AWSSupport-ManageRDPSettings to manage the RDP settings.
4. (Offline remediation) If the instance is not a managed instance then:
   a. aws:assertAwsResourceProperty - Assert AllowOffline = True
   b. aws:assertAwsResourceProperty - Assert Action = FixAll
c. aws:assertAwsResourceProperty - Assert the value of SubnetId
   (Use the provided instance's subnet) If SubnetId is SELECTED_INSTANCE_SUBNET
   aws:executeAwsApi - Retrieve the current instance's subnet.
   aws:executeAutomation - Run AWSSupport-ExecuteEC2Rescue with provided instance's subnet.
d. (Use the provided custom subnet) If SubnetId is not SELECTED_INSTANCE_SUBNET
   aws:executeAutomation - Run AWSSupport-ExecuteEC2Rescue with provided SubnetId value.

Outputs

manageFirewallProfiles.Output
manageRDPServiceSettings.Output
manageRDPSettings.Output
checkFirewallProfiles.Output
checkRDPServiceSettings.Output
checkRDPSettings.Output
disableFirewallProfiles.Output
restoreDefaultRDPServiceSettings.Output
restoreDefaultRDPSettings.Output
troubleshootRDPOffline.Output
troubleshootRDPOfflineWithSubnetId.Output

AWSSupport-TroubleshootSSH

Description

The AWSSupport-TroubleshootSSH automation document installs the Amazon EC2Rescue tool for Linux, and then uses the EC2Rescue tool to check or attempt to fix common issues that prevent a remote connection to the Linux machine via SSH. Optionally, changes can be applied offline by stopping and starting the instance, if the user explicitly allows for offline remediation. By default, the document operates in read-only mode.

Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Linux

Parameters
• InstanceId
  Type: String
  Description: (Required) ID of your EC2 instance for Linux.
• Action
  Type: String
  Allowed values: CheckAll, FixAll
  Default: CheckAll
  Description: (Required) Specify whether to check for issues without fixing them or to check and automatically fix any discovered issues.
• AllowOffline
  Type: String
  Allowed values: True, False
  Default: False
  Description: (Optional) Fix only - Set it to true if you allow an offline SSH remediation in case the online troubleshooting fails, or the provided instance is not a managed instance. Note: For the offline remediation, SSM Automation stops the instance, and creates an AMI before attempting any operations.
• SubnetId
  Type: String
  Default: SelectedInstanceSubnet
  Description: (Optional) Offline only - The subnet ID for the EC2Rescue instance used to perform the offline troubleshooting. If no subnet ID is specified, AWS Systems Manager Automation will create a new VPC.

  **Important**
  The subnet must be in the same Availability Zone as InstanceId, and it must allow access to the SSM endpoints.
• S3BucketName
  Type: String
  Description: (Optional) Offline only - S3 bucket name in your account where you want to upload the troubleshooting logs. Make sure the bucket policy does not grant unnecessary read/write permissions to parties that do not need access to the collected logs.
• AutomationAssumeRole
  Type: String
  Description: (Optional) The IAM role for this execution. If no role is specified, Systems Manager Automation will use the permissions of the user that runs this document.

**Required IAM Permissions**

It is recommended that the EC2 instance receiving the command has an IAM role with the [AmazonSSMManagedInstanceCore](https://docs.aws.amazon.com/ssm/latest/userguide/iam-role.html) Amazon managed policy attached. For the online remediation, the user must have at least [ssm:DescribeInstanceInformation](https://docs.aws.amazon.com/ssm/latest/userguide/iam-role.html#ssm-describe-instance-information), [ssm:ExecuteAutomation](https://docs.aws.amazon.com/ssm/latest/userguide/iam-role.html#ssm-execute-automation)
and `ssm:SendCommand` to run the automation and send the command to the instance, plus `ssm:GetAutomationExecution` to be able to read the automation output. For the offline remediation, the user must have at least `ssm:DescribeInstanceInformation`, `ssm:ExecuteAutomation`, `ec2:DescribeInstances`, plus `ssm:GetAutomationExecution` to be able to read the automation output. AWSSupport-TroubleshootSSH calls AWSSupport-ExecuteEC2Rescue to perform the offline remediation - please review the permissions for AWSSupport-ExecuteEC2Rescue to ensure you can run the automation successfully.

**Document Steps**

1. `aws:assertAwsResourceProperty` - Check if the instance is a managed instance
   a. (Online remediation) If the instance is a managed instance, then:
      i. `aws:configurePackage` - Install EC2Rescue for Linux via AWS-ConfigureAWSPackage.
      ii. `aws:runCommand` - Run the bash script to run EC2Rescue for Linux.
   b. (Offline remediation) If the instance is not a managed instance then:
      i. `aws:assertAwsResourceProperty` - Assert `AllowOffline = True`
      ii. `aws:assertAwsResourceProperty` - Assert `Action = FixAll`
      iii. `aws:assertAwsResourceProperty` - Assert the value of SubnetId
      iv. (Use the provided instance's subnet) If SubnetId is SelectedInstanceSubnet use `aws:executeAutomation` to run AWSSupport-ExecuteEC2Rescue with provided instance's subnet.
      v. (Use the provided custom subnet) If SubnetId is not SelectedInstanceSubnet use `aws:executeAutomation` to run AWSSupport-ExecuteEC2Rescue with provided SubnetId value.

**Outputs**

- troubleshootSSH.Output
- troubleshootSSHOffline.Output
- troubleshootSSHOfflineWithSubnetId.Output

**AWS-UpdateCloudFormationStack**

**Description**

Update an AWS CloudFormation stack by using an AWS CloudFormation template stored in an Amazon S3 bucket.

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows, Linux

**Parameters**

- AutomationAssumeRole
  Type: String
AWS-UpdateLinuxAmi

Description
Update an Amazon Machine Image (AMI) with Linux distribution packages and Amazon software.

Run this Automation (console)

Document Type
Automation

Owner
Amazon

Platforms
Windows, Linux

Parameters
• AutomationAssumeRole
  Type: String
  Default: arn:aws:iam::{{global:ACCOUNT_ID}}:role/AutomationServiceRole
  Description: (Required) The ARN of the role that allows Automation to perform the actions on your behalf.

• ExcludePackages
  Type: String
  Default: none
  Description: (Optional) Names of packages to hold back from updates, under all conditions. By default ("none"), no package is excluded.

• IamInstanceProfileName
Type: String
Default: ManagedInstanceProfile
Description: (Required) The instance profile that enables Systems Manager to manage the instance.
- IncludePackages
  Type: String
  Default: all
  Description: (Optional) Only update these named packages. By default ("all"), all available updates are applied.
- InstanceType
  Type: String
  Default: t2.micro
  Description: (Optional) Type of instance to launch as the workspace host. Instance types vary by Region.
- PostUpdateScript
  Type: String
  Default: none
  Description: (Optional) URL of a script to run after package updates are applied. Default ("none") is to not run a script.
- PreUpdateScript
  Type: String
  Default: none
  Description: (Optional) URL of a script to run before updates are applied. Default ("none") is to not run a script.
- SourceAmiId
  Type: String
  Description: (Required) The source Amazon Machine Image ID.
- TargetAmiName
  Type: String
  Default: UpdateLinuxAmi_from_{{SourceAmiId}}_on_{{global:DATE_TIME}}
  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.

AWS-UpdateWindowsAmi

Description
Update a Microsoft Windows Amazon Machine Image (AMI). By default, this document installs all Windows updates, Amazon software, and Amazon drivers. It then runs Sysprep to create a new AMI. Supports Windows Server 2008 R2 or later.
Run this Automation (console)

Document Type

Automation

Owner

Amazon

Platforms

Windows, Linux

Parameters

• AutomationAssumeRole
  Type: String
  Default: arn:aws:iam::{{global:ACCOUNT_ID}}:role/AutomationServiceRole
  Description: (Required) The ARN of the role that allows Automation to perform the actions on your behalf.

• Categories
  Type: String
  Description: (Optional) Specify one or more update categories. You can filter categories using comma-separated values. Options: Application, Connectors, CriticalUpdates, DefinitionUpdates, DeveloperKits, Drivers, Feature Packs, Guidance, Microsoft, Security Updates, Service Packs, Tools, Update Rollups, Updates. Valid formats include a single entry, for example: CriticalUpdates. Or you can specify a comma separated list: CriticalUpdates,SecurityUpdates. NOTE: There cannot be any spaces around the commas.

• 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. Valid formats: KB9876543 or 9876543.

• IamInstanceProfileName
  Type: String
  Default: ManagedInstanceProfile
  Description: (Required) The name of the role that enables Systems Manager to manage the instance.

• 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. Valid formats: KB9876543 or 9876543.

• InstanceType
  Type: String
  Default: t2.medium
Description: (Optional) Type of instance to launch as the workspace host. Instance types vary by region. Default is t2.medium.

- **PostUpdateScript**
  Type: String
  Description: (Optional) A script provided as a string. It will run after installing OS updates.

- **PreUpdateScript**
  Type: String
  Description: (Optional) A script provided as a string. It will run prior to installing OS updates.

- **PublishedDateAfter**
  Type: String
  Description: (Optional) Specify the date that the updates should be published after. For example, if 01/01/2017 is specified, any updates that were found during the Windows Update search that have been published on or after 01/01/2017 will be returned.

- **PublishedDateBefore**
  Type: String
  Description: (Optional) Specify the date that the updates should be published before. For example, if 01/01/2017 is specified, any updates that were found during the Windows Update search that have been published on or before 01/01/2017 will be returned.

- **PublishedDaysOld**
  Type: String
  Description: (Optional) Specify the amount of days old the updates must be from the published date. For example, if 10 is specified, any updates that were found during the Windows Update search that have been published 10 or more days ago will be returned.

- **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.

- **SourceAmiId**
  Type: String
  Description: (Required) The source Amazon Machine Image ID.

- **SubnetId**
  Type: String
  Description: (Optional) Specify the SubnetId if you want to launch into a specific subnet.

- **TargetAmiName**
  Type: String
  Default: `UpdateWindowsAmi_from_{{SourceAmiId}}_on_{{global:DATE_TIME}}`
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.

**AWSSupport-UpgradeWindowsAWSDrivers**

**Description**

The AWSSupport-UpgradeWindowsAWSDrivers upgrades or repairs storage and network AWS drivers on the specified EC2 instance. The document attempts to install the latest versions of AWS drivers online by calling SSM Agent. If SSM Agent is not contactable, the document can perform an offline installation of the AWS drivers if explicitly requested. Note: Both the online and offline upgrade will create an AMI before attempting any operations, which will persist after the automation completes. It is your responsibility to secure access to the AMI, or to delete it. The online method restarts the instance as part of the upgrade process, while the offline method requires the provided EC2 instance be stopped and then started.

**Important**

If your instances connect to AWS Systems Manager using VPC endpoints, this document will fail unless used in the us-east-1 Region. This document will also fail on a domain controller. To update AWS PV drivers on a domain controller, see Upgrade a Domain Controller (AWS PV Upgrade).

**Run this Automation (console)**

**Document Type**

Automation

**Owner**

Amazon

**Platforms**

Windows

**Parameters**

- **InstanceId**
  
  Type: String
  
  Description: (Required) ID of your EC2 instance for Windows Server.

- **AllowOffline**
  
  Type: String
  
  Allowed values: True,False
  
  Default: False
  
  Description: (Optional) Set it to true if you allow an offline drivers upgrade in case the online installation cannot be performed. Note: The offline method requires the provided EC2 instance be stopped and then started. Data stored in instance store volumes will be lost. The public IP address will change if you are not using an Elastic IP.

- **SubnetId**
  
  Type: String
  
  Default: SelectedInstanceSubnet
Description: (Optional) Offline only - The subnet ID for the EC2Rescue instance used to perform the offline drivers upgrade. If no subnet ID is specified, Systems Manager Automation will create a new VPC.

**Important**
The subnet must be in the same Availability Zone as InstanceId, and it must allow access to the SSM endpoints.

- **ForceUpgrade**

  Type: String

  Allowed values: True, False

  Default: False

  Description: (Optional) Offline only - Set it to true if you allow the offline drivers upgrade to proceed even though your instance already has the latest drivers installed.

- **AutomationAssumeRole**

  Type: String

  Description: (Optional) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to run this document.

**Required IAM Permissions**

The EC2 instance receiving the command must at minimum have an IAM role that includes permissions for `ssm:StartAutomationExecution` and `ssm:SendCommand` to run the automation and send the command to the instance, plus `ssm:GetAutomationExecution` to be able to read the automation output. You can attach the `AmazonSSMManagedInstanceCore` Amazon managed policy to your IAM role to provide these permissions. We recommend, however, using the Automation IAM role `AmazonSSMAutomationRole` for this purpose. For more information, see Method 2: Use IAM to configure roles for Automation (p. 283).

If you are performing an offline upgrade, see the permissions required by `AWSSupport-StartEC2RescueWorkflow` (p. 512).

**Document Steps**

1. **aws:assertAwsResourceProperty** - Verifies the input instance is Windows.
2. **aws:assertAwsResourceProperty** - Verifies the input instance is a managed instance. If so, the online upgrade starts, otherwise the offline upgrade is evaluated.
   a. (Online upgrade) If the input instance is a managed instance:
      i. **aws:createImage** - Creates an AMI backup.
      ii. **aws:createTags** - Tags the AMI backup.
      iii. **aws:runCommand** - Installs ENA network driver via AWS-ConfigureAWSPackage.
     iv. **aws:runCommand** - Installs NVMe driver via AWS-ConfigureAWSPackage.
     v. **aws:runCommand** - Installs AWS PV driver via AWS-ConfigureAWSPackage.
   b. (Offline upgrade) If the input instance is not a managed instance:
      i. **aws:assertAwsResourceProperty** - Verifies the AllowOffline flag is set to True. If so, the offline upgrade starts, otherwise the workflow ends.
      ii. **aws:changeInstanceState** - Stop the source instance.
      iii. **aws:changeInstanceState** - Force-stop the source instance.
iv. `aws:createImage` - Create an AMI backup of the source instance.
v. `aws:createTags` - Tag the AMI backup of the source instance.
vi. `aws:executeAwsApi` - Enable ENA for the instance
viii. **(Force offline upgrade)** If `ForceUpgrade = True` then run `aws:executeAutomation` to invoke `AWSSupport-StartEC2RescueWorkflow` with the drivers force upgrade script. This installs the drivers regardless of the current version that is installed.
ix. **(Offline upgrade)** If `ForceUpgrade = False` then run `aws:executeAutomation` to invoke `AWSSupport-StartEC2RescueWorkflow` with the drivers upgrade script.

Outputs

- `preUpgradeBackup.ImageId`
- `preOfflineUpgradeBackup.ImageId`
- `installAwsEnaNetworkDriverOnInstance.Output`
- `installAWSNVMeOnInstance.Output`
- `installAWSPVDriverOnInstance.Output`
- `upgradeDriversOffline.Output`
- `forceUpgradeDriversOffline.Output`

**Sample scenarios and custom Automation document solutions**

The following sample Automation documents demonstrate how you can use AWS Systems Manager Automation actions to automate common deployment, troubleshooting, and maintenance tasks.

**Note**

The Automation document samples in this section are provided to demonstrate how you can create custom Automation documents to support your specific operational needs. These documents are not meant for use in production environments as is. However, you can customize them for your own use.

**Samples**

- Deploy VPC architecture and Microsoft Active Directory domain controllers (p. 539)
- Restore a root volume from the latest snapshot (p. 558)
- Create an AMI and cross-Region copy (p. 566)

**Deploy VPC architecture and Microsoft Active Directory domain controllers**

To increase efficiency and standardize common tasks, you might choose to automate deployments. This is useful if you regularly deploy the same architecture across multiple accounts and Regions. Automating architecture deployments can also reduce the potential for human error that can occur when deploying architecture manually. AWS Systems Manager Automation actions can help you accomplish this.

The following sample AWS Systems Manager Automation document performs these actions.

- Retrieves the latest Windows Server 2012R2 Amazon Machine Image (AMI) using Systems Manager Parameter Store to use when launching the EC2 instances that will be configured as domain controllers.
- Uses the `aws:executeAwsApi` Automation action to call several AWS API actions to create the VPC architecture. The domain controller instances are launched in private subnets, and connect to the...
internet using a NAT gateway. This enables the SSM Agent on the instances to access the requisite Systems Manager endpoints.

- Uses the `aws:waitForAwsResourceProperty` Automation action to confirm the instances launched by the previous action are Online for AWS Systems Manager.
- Uses the `aws:runCommand` Automation action to configure the instances launched as Microsoft Active Directory domain controllers.

**YAML**

```yaml
---
description: Custom Automation Deployment Sample
schemaVersion: '0.3'
parameters:
  AutomationAssumeRole:
    type: String
    default: ''
    description: >-
      (Optional) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to execute this document.
mainSteps:
- name: getLatestWindowsAmi
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ssm
    Api: GetParameter
    Name: >-
      /aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-English-64Bit-Base
  outputs:
    - Name: amiId
      Selector: $.Parameter.Value
      Type: String
    nextStep: createSSMInstanceRole
- name: createSSMInstanceRole
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: iam
    Api: CreateRole
    AssumeRolePolicyDocument: >-
      "Version": "2012-10-17","Statement": [{"Effect": "Allow","Principal": {"Service": ["ec2.amazonaws.com"]},"Action": ["sts:AssumeRole"]}]"
    RoleName: sampleSSMInstanceRole
    nextStep: attachManagedSSMPolicy
- name: attachManagedSSMPolicy
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: iam
    Api: AttachRolePolicy
    PolicyArn: 'arn:aws:iam::aws:policy/service-role/AmazonSSMManagedInstanceCore'
    RoleName: sampleSSMInstanceRole
    nextStep: createSSMInstanceProfile
- name: createSSMInstanceProfile
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: iam
    Api: CreateInstanceProfile
    InstanceProfileName: sampleSSMInstanceRole
  outputs:
```

540
- Name: instanceProfileArn
  Selector: $.InstanceProfile.Arn
  Type: String
  nextStep: addSSMInstanceRoleToProfile
- name: addSSMInstanceRoleToProfile
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: iam
    Api: AddRoleToInstanceProfile
    InstanceProfileName: sampleSSMInstanceRole
    RoleName: sampleSSMInstanceRole
  nextStep: createVpc
- name: createVpc
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateVpc
    CidrBlock: 10.0.100.0/22
  outputs:
    - Name: vpcId
      Selector: $.Vpc.VpcId
      Type: String
  nextStep: getMainRtb
- name: getMainRtb
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeRouteTables
    Filters:
      - Name: vpc-id
        Values:
        - '{{ createVpc.vpcId }}'
  outputs:
    - Name: mainRtbId
      Selector: $.RouteTables[0].RouteTableId
      Type: String
  nextStep: verifyMainRtb
- name: verifyMainRtb
  action: aws:assertAwsResourceProperty
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeRouteTables
    RouteTableIds:
    - '{{ getMainRtb.mainRtbId }}'
    PropertySelector: '$.RouteTables[0].Associations[0].Main'
    DesiredValues:
    - 'True'
  nextStep: createPubSubnet
- name: createPubSubnet
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateSubnet
    CidrBlock: 10.0.103.0/24
    AvailabilityZone: us-west-2c
    VpcId: '{{ createVpc.vpcId }}'
  outputs:
    - Name: pubSubnetId
      Selector: $.Subnet.SubnetId
      Type: String
  nextStep: createPubRtb
- name: createPubRtb
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateRouteTable
    VpcId: '{{ createVpc.vpcId }}'
  outputs:
    - Name: pubRtbId
      Selector: $.RouteTable.RouteTableId
      Type: String
  nextStep: createIgw

- name: createIgw
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateInternetGateway
  outputs:
    - Name: igwId
      Selector: $.InternetGateway.InternetGatewayId
      Type: String
  nextStep: attachIgw

- name: attachIgw
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: AttachInternetGateway
    InternetGatewayId: '{{ createIgw.igwId }}'
    VpcId: '{{ createVpc.vpcId }}'
  nextStep: allocateEip

- name: allocateEip
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: AllocateAddress
    Domain: vpc
  outputs:
    - Name: eipAllocationId
      Selector: $.AllocationId
      Type: String
  nextStep: createNatGw

- name: createNatGw
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateNatGateway
    AllocationId: '{{ allocateEip.eipAllocationId }}'
    SubnetId: '{{ createPubSubnet.pubSubnetId }}'
  outputs:
    - Name: natGwId
      Selector: $.NatGateway.NatGatewayId
      Type: String
  nextStep: verifyNatGwAvailable

- name: verifyNatGwAvailable
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 150
  inputs:
    Service: ec2
    Api: DescribeNatGateways
    NatGatewayIds:
      - '{{ createNatGw.natGwId }}'
    PropertySelector: '$.NatGateways[0].State'
DesiredValues:
  - available
nextStep: createNatRoute
- name: createNatRoute
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateRoute
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: '{{ createNatGw.natGwId }}'
    RouteTableId: '{{ getMainRtb.mainRtbId }}'
nextStep: createPubRoute
- name: createPubRoute
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateRoute
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: '{{ createIgw.igwId }}'
    RouteTableId: '{{ createPubRtb.pubRtbId }}'
nextStep: setPubSubAssoc
- name: setPubSubAssoc
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: AssociateRouteTable
    RouteTableId: '{{ createPubRtb.pubRtbId }}'
    SubnetId: '{{ createPubSubnet.pubSubnetId }}'
- name: createDhcpOptions
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateDhcpOptions
    DhcpConfigurations:
      - Key: domain-name-servers
        Values:
        - '10.0.100.50,10.0.101.50'
      - Key: domain-name
        Values:
        - sample.com
  outputs:
    - Name: dhcpOptionsId
      Selector: $.DhcpOptions.DhcpOptionsId
      Type: String
nextStep: createDCSubnet1
- name: createDCSubnet1
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateSubnet
    CidrBlock: 10.0.100.0/24
    AvailabilityZone: us-west-2a
    VpcId: '{{ createVpc.vpcId }}'
  outputs:
    - Name: firstSubnetId
      Selector: $.Subnet.SubnetId
      Type: String
nextStep: createDCSubnet2
- name: createDCSubnet2
  action: aws:executeAwsApi
  onFailure: Abort
inputs:
  Service: ec2
  Api: CreateSubnet
  CidrBlock: 10.0.101.0/24
  AvailabilityZone: us-west-2b
  VpcId: '{{ createVpc.vpcId }}'
outputs:
  - Name: secondSubnetId
    Selector: $.Subnet.SubnetId
    Type: String
nextStep: createDCSecGroup
- name: createDCSecGroup
  action: aws:executeAwsApi
  onFailure: Abort
inputs:
  Service: ec2
  Api: CreateSecurityGroup
  GroupName: SampleDCSecGroup
  Description: Security Group for Sample Domain Controllers
  VpcId: '{{ createVpc.vpcId }}'
outputs:
  - Name: dcSecGroupId
    Selector: $.GroupId
    Type: String
nextStep: authIngressDCTraffic
- name: authIngressDCTraffic
  action: aws:executeAwsApi
  onFailure: Abort
inputs:
  Service: ec2
  Api: AuthorizeSecurityGroupIngress
  GroupId: '{{ createDCSecGroup.dcSecGroupId }}'
  IpPermissions:
    - FromPort: -1
    IpProtocol: '-1'
  IpRanges:
    - CidrIp: 0.0.0.0/0
    Description: Allow all traffic between Domain Controllers
nextStep: verifyInstanceProfile
- name: verifyInstanceProfile
  action: aws:waitForAwsResourceProperty
  maxAttempts: 5
  onFailure: Abort
inputs:
  Service: iam
  Api: ListInstanceProfilesForRole
  RoleName: sampleSSMInstanceRole
  PropertySelector: '$.InstanceProfiles[0].Arn'
  DesiredValues:
    - '{{ createSSMInstanceProfile.instanceProfileArn }}'
nextStep: iamEventualConsistency
- name: iamEventualConsistency
  action: aws:sleep
  inputs:
    Duration: PT2M
nextStep: launchDC1
- name: launchDC1
  action: aws:executeAwsApi
  onFailure: Abort
inputs:
  Service: ec2
  Api: RunInstances
  BlockDeviceMappings:
    - DeviceName: /dev/sda1
      Ebs:
        DeleteOnTermination: true
VolumeSize: 50
VolumeType: gp2
- DeviceName: xvdf
  Ebs:
  DeleteOnTermination: true
  VolumeSize: 100
  VolumeType: gp2
IamInstanceProfile:
  Arn: '{{ createSSMInstanceProfile.instanceProfileArn }}'
  ImageId: '{{ getLatestWindowsAmi.amiId }}'
InstanceType: t2.micro
MaxCount: 1
MinCount: 1
PrivateIpAddress: 10.0.100.50
SecurityGroupIds:
- '{{ createDCSecGroup dcSecGroupId }}'
SubnetId: '{{ createDCSubnet1.firstSubnetId }}'
TagSpecifications:
  - ResourceType: instance
    Tags:
      - Key: Name
        Value: SampleDC1
outputs:
  - Name: pdcInstanceId
    Selector: '$.Instances[0].InstanceId'
nextStep: launchDC2
- name: launchDC2
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: RunInstances
    BlockDeviceMappings:
      - DeviceName: /dev/sda1
        Ebs:
          DeleteOnTermination: true
          VolumeSize: 50
          VolumeType: gp2
      - DeviceName: xvdf
        Ebs:
          DeleteOnTermination: true
          VolumeSize: 100
          VolumeType: gp2
IamInstanceProfile:
  Arn: '{{ createSSMInstanceProfile instanceProfileArn }}'
  ImageId: '{{ getLatestWindowsAmi.amiId }}'
InstanceType: t2.micro
MaxCount: 1
MinCount: 1
PrivateIpAddress: 10.0.101.50
SecurityGroupIds:
- '{{ createDCSecGroup dcSecGroupId }}'
SubnetId: '{{ createDCSubnet2.secondSubnetId }}'
TagSpecifications:
  - ResourceType: instance
    Tags:
      - Key: Name
        Value: SampleDC2
outputs:
  - Name: adcInstanceId
    Selector: '$.Instances[0].InstanceId'
nextStep: verifyDCInstanceState
- name: verifyDCInstanceState
  action: aws:waitForAwsResourceProperty
inputs:
  Service: ec2
  Api: DescribeInstanceStatus
  IncludeAllInstances: true
  InstanceIds:
  - '{ launchDC1.pdcInstanceId }'
  - '{ launchDC2.adcInstanceId }'
  PropertySelector: '.InstanceStatuses[0].InstanceState.Name'
  DesiredValues:
  - running
  nextStep: verifyInstancesOnlineSSM
  - name: verifyInstancesOnlineSSM
    action: aws:waitForAwsResourceProperty
    timeoutSeconds: 600
  inputs:
    Service: ssm
    Api: DescribeInstanceInformation
    InstanceInformationFilterList:
    - key: InstanceIds
      valueSet:
      - '{ launchDC1.pdcInstanceId }'
      - '{ launchDC2.adcInstanceId }'
      PropertySelector: '.InstanceInformationList[0].PingStatus'
    DesiredValues:
    - Online
    nextStep: installADRoles
  - name: installADRoles
    action: aws:runCommand
    inputs:
      DocumentName: AWS-RunPowerShellScript
      InstanceIds:
      - '{ launchDC1.pdcInstanceId }'
      - '{ launchDC2.adcInstanceId }'
      Parameters:
        commands: |-
          try {
            Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
          }
          catch {
            Write-Error "Failed to install ADDS Role."
          }
        nextStep: setAdminPassword
  - name: setAdminPassword
    action: aws:runCommand
    inputs:
      DocumentName: AWS-RunPowerShellScript
      InstanceIds:
      - '{ launchDC1.pdcInstanceId }'
      Parameters:
        commands: |-
          net user Administrator "sampleAdminPass123!"
        nextStep: createForest
  - name: createForest
    action: aws:runCommand
    inputs:
      DocumentName: AWS-RunPowerShellScript
      InstanceIds:
      - '{ launchDC1.pdcInstanceId }'
      Parameters:
        commands: |-
          $dsrmPass = 'sample123!' | ConvertTo-SecureString -asPlainText -Force
          try {
            Install-ADDSForest -DomainName "sample.com" -DomainMode 6 -ForestMode 6 -InstallDNS -DatabasePath "D:\NTDS" -SysvolPath "D:\SYSVOL" -SafeModeAdministratorPassword $dsrmPass -Force
          }
catch {
    Write-Error $
} 
try {
    Add-DnsServerForwarder -IPAddress "10.0.100.2"
} 
catch {
    Write-Error $
} 

nextStep: associateDhcpOptions
- name: associateDhcpOptions
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: AssociateDhcpOptions
    DhcpOptionsId: '{{ createDhcpOptions.dhcpOptionsId }}'
    VpcId: '{{ createVpc.vpcId }}'

nextStep: waitForADServices
- name: waitForADServices
  action: aws:sleep
  inputs:
    Duration: PT1M

nextStep: promoteADC
- name: promoteADC
  action: aws:runCommand
  inputs:
    DocumentName: AWS-RunPowerShellScript
    InstanceIds:
      - '{( launchDC2.adcInstanceId )}'
    commands: |
      $dsrmPass = 'sample123!' | ConvertTo-SecureString -asPlainText -Force
      $domAdminUser = "sample\Administrator"
      $domAdminPass = "sampleAdminPass123!" | ConvertTo-SecureString -asPlainText -Force
      try {
        Install-ADDSDomainController -DomainName "sample.com" -InstallDNS -DatabasePath "D:\NTDS" -SysvolPath "D:\SYSVOL" -SafeModeAdministratorPassword $dsrmPass -Credential $domAdminCred -Force
      } 
catch {
        Write-Error $
      } 

JSON

```json
{
    "description": "Custom Automation Deployment Sample",
    "schemaVersion": "0.3",
    "assumeRole": "{AutomationAssumeRole }",
    "parameters": {
        "AutomationAssumeRole": {
            "type": "String",
            "description": "(Optional) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to execute this document."
        }
    },
}
```
"mainSteps": [
    {
        "name": "getLatestWindowsAmi",
        "action": "aws:executeAwsApi",
        "onFailure": "Abort",
        "inputs": {
            "Service": "ssm",
            "Api": "GetParameter",
            "Name": "/aws/service/ami-windows-latest/Windows_Server-2012-R2_RTM-English-64Bit-Base"
        },
        "outputs": [
            {
                "Name": "amiId",
                "Selector": ".$.Parameter.Value",
                "Type": "String"
            }
        ],
        "nextStep": "createSSMInstanceRole"
    },
    {
        "name": "createSSMInstanceRole",
        "action": "aws:executeAwsApi",
        "onFailure": "Abort",
        "inputs": {
            "Service": "iam",
            "Api": "CreateRole",
            "AssumeRolePolicyDocument": "{"Version":"2012-10-17","Statement": [{"Effect":"Allow","Principal":{"Service":["ec2.amazonaws.com"]},"Action": ["sts:AssumeRole"]}],
            "RoleName": "sampleSSMInstanceRole"
        },
        "nextStep": "attachManagedSSMPolicy"
    },
    {
        "name": "attachManagedSSMPolicy",
        "action": "aws:executeAwsApi",
        "onFailure": "Abort",
        "inputs": {
            "Service": "iam",
            "Api": "AttachRolePolicy",
            "PolicyArn": "arn:aws:iam::aws:policy/service-role/AmazonSSMManagedInstanceCore",
            "RoleName": "sampleSSMInstanceRole"
        },
        "nextStep": "createSSMInstanceProfile"
    },
    {
        "name": "createSSMInstanceProfile",
        "action": "aws:executeAwsApi",
        "onFailure": "Abort",
        "inputs": {
            "Service": "iam",
            "Api": "CreateInstanceProfile",
            "InstanceProfileName": "sampleSSMInstanceRole"
        },
        "outputs": [
            {
                "Name": "instanceProfileArn",
                "Selector": ".$.InstanceProfile.Arn",
                "Type": "String"
            }
        ],
        "nextStep": "addSSMInstanceRoleToProfile"
    }
]
"name": "addSSMInstanceRoleToProfile",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
  "Service": "iam",
  "Api": "AddRoleToInstanceProfile",
  "InstanceProfileName": "sampleSSMInstanceRole",
  "RoleName": "sampleSSMInstanceRole"
},
"nextStep": "createVpc"
},
{
  "name": "createVpc",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateVpc",
    "CidrBlock": "10.0.100.0/22"
  },
  "outputs": [
    {
      "Name": "vpcId",
      "Selector": "$.Vpc.VpcId",
      "Type": "String"
    }
  ],
  "nextStep": "getMainRtb"
},
{
  "name": "getMainRtb",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "DescribeRouteTables",
    "Filters": [
      {
        "Name": "vpc-id",
        "Values": ["{{ createVpc.vpcId }}"]
      }
    ]
  },
  "outputs": [
    {
      "Name": "mainRtbId",
      "Selector": "$.RouteTables[0].RouteTableId",
      "Type": "String"
    }
  ],
  "nextStep": "verifyMainRtb"
},
{
  "name": "verifyMainRtb",
  "action": "aws:assertAwsResourceProperty",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "DescribeRouteTables",
    "RouteTableIds": ["{{ getMainRtb.mainRtbId }}"],
    "PropertySelector": "$.RouteTables[0].Associations[0].Main",
    "DesiredValues": ["True"]
  },
  "nextStep": "createPubSubnet"
}
"name": "createPubSubnet",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
  "Service": "ec2",
  "Api": "CreateSubnet",
  "CidrBlock": "10.0.103.0/24",
  "AvailabilityZone": "us-west-2c",
  "VpcId": "{{ createVpc.vpcId }}"
},
"outputs": [
  {
    "Name": "pubSubnetId",
    "Selector": "$.Subnet.SubnetId",
    "Type": "String"
  }
],
"nextStep": "createPubRtb"
},

{ "name": "createPubRtb",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateRouteTable",
    "VpcId": "{{ createVpc.vpcId }}"
  },
  "outputs": [
    {
      "Name": "pubRtbId",
      "Selector": "$.RouteTable.RouteTableId",
      "Type": "String"
    }
  ],
  "nextStep": "createIgw"
},

{ "name": "createIgw",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateInternetGateway"
  },
  "outputs": [
    {
      "Name": "igwId",
      "Selector": "$.InternetGateway.InternetGatewayId",
      "Type": "String"
    }
  ],
  "nextStep": "attachIgw"
},

{ "name": "attachIgw",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "AttachInternetGateway",
    "InternetGatewayId": "{{ createIgw.igwId }}",
    "VpcId": "{{ createVpc.vpcId }}"
  },
  "nextStep": "allocateEip"
},
```json
{
  "name": "allocateEip",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "AllocateAddress",
    "Domain": "vpc"
  },
  "outputs": [
    {
      "Name": "eipAllocationId",
      "Selector": "$.AllocationId",
      "Type": "String"
    }
  ],
  "nextStep": "createNatGw"
},
{
  "name": "createNatGw",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateNatGateway",
    "AllocationId": "{{ allocateEip.eipAllocationId }}",
    "SubnetId": "{{ createPubSubnet.pubSubnetId }}"
  },
  "outputs": [
    {
      "Name": "natGwId",
      "Selector": "$.NatGateway.NatGatewayId",
      "Type": "String"
    }
  ],
  "nextStep": "verifyNatGwAvailable"
},
{
  "name": "verifyNatGwAvailable",
  "action": "aws:waitForAwsResourceProperty",
  "timeoutSeconds": 150,
  "inputs": {
    "Service": "ec2",
    "Api": "DescribeNatGateways",
    "NatGatewayIds": [
      "{{ createNatGw.natGwId }}"
    ],
    "PropertySelector": "$.NatGateways[0].State",
    "DesiredValues": ["available"
  ],
  "nextStep": "createNatRoute"
},
{
  "name": "createNatRoute",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateRoute",
    "DestinationCidrBlock": "0.0.0.0/0",
    "NatGatewayId": "{{ createNatGw.natGwId }}",
    "RouteTableId": "{{ getMainRtb.mainRtbId }}"
  },
  "nextStep": "createPubRoute"
}
```
{},
{
  "name": "createPubRoute",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateRoute",
    "DestinationCidrBlock": "0.0.0.0/0",
    "GatewayId": "{{ createIgw.igwId }}",
    "RouteTableId": "{{ createPubRtb.pubRtbId }}"
  },
  "nextStep": "setPubSubAssoc"
},
{
  "name": "setPubSubAssoc",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "AssociateRouteTable",
    "RouteTableId": "{{ createPubRtb.pubRtbId }}",
    "SubnetId": "{{ createPubSubnet.pubSubnetId }}"
  }
},
{
  "name": "createDhcpOptions",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateDhcpOptions",
    "DhcpConfigurations": [
      {
        "Key": "domain-name-servers",
        "Values": ["10.0.100.50,10.0.101.50"]
      },
      {
        "Key": "domain-name",
        "Values": ["sample.com"]
      }
    ]
  },
  "outputs": [
    {
      "Name": "dhcpOptionsId",
      "Selector": ".DhcpOptions.DhcpOptionsId",
      "Type": "String"
    }
  ],
  "nextStep": "createDCSubnet1"
},
{
  "name": "createDCSubnet1",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "CreateSubnet",
    "CidrBlock": "10.0.100.0/24",
    "AvailabilityZone": "us-west-2a",
    "VpcId": "{{ createVpc.vpcId }}"
  },
  "outputs": [
    {
      "Name": "firstSubnetId",
      "Selector": ".Subnets[0].SubnetId",
      "Type": "String"
    }
  ]
}
"Selector": ".Subnet.SubnetId",
"Type": "String"
}
],
"nextStep": "createDCSubnet2"
},
{
"name": "createDCSubnet2",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
"Service": "ec2",
"Api": "CreateSubnet",
"CidrBlock": "10.0.101.0/24",
"AvailabilityZone": "us-west-2b",
"VpcId": "{{ createVpc.vpcId }}"
},
"outputs": [
{
"Name": "secondSubnetId",
"Selector": ".Subnet.SubnetId",
"Type": "String"
}
],
"nextStep": "createDCSecGroup"
},
{
"name": "createDCSecGroup",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
"Service": "ec2",
"Api": "CreateSecurityGroup",
"GroupName": "SampleDCSecGroup",
"Description": "Security Group for Example Domain Controllers",
"VpcId": "{{ createVpc.vpcId }}"
},
"outputs": [
{
"Name": "dcSecGroupId",
"Selector": ".GroupId",
"Type": "String"
}
],
"nextStep": "authIngressDCTraffic"
},
{
"name": "authIngressDCTraffic",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
"Service": "ec2",
"Api": "AuthorizeSecurityGroupIngress",
"GroupId": "{{ createDCSecGroup.dcSecGroupId }}",
"IpPermissions": [
{
"FromPort": -1,
"IpProtocol": "-1",
"IpRanges": [
{
"CidrIp": "0.0.0.0/0",
"Description": "Allow all traffic between Domain Controllers"
}
]
}
]
{
  "name": "createSSMInstanceProfile",
  "action": "aws:waitForAwsResourceProperty",
  "maxAttempts": 5,
  "onFailure": "Abort",
  "inputs": {
    "Service": "iam",
    "Api": "ListInstanceProfilesForRole",
    "RoleName": "sampleSSMInstanceRole",
    "PropertySelector": "$.InstanceProfiles[0].Arn",
    "DesiredValues": [
      "{{ createSSMInstanceProfile.instanceProfileArn }}"
    ]
  },
  "nextStep": "iamEventualConsistency"
},
{
  "name": "iamEventualConsistency",
  "action": "aws:sleep",
  "inputs": {
    "Duration": "PT2M"
  },
  "nextStep": "launchDC1"
},
{
  "name": "launchDC1",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "RunInstances",
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/sda1",
        "Ebs": {
          "DeleteOnTermination": true,
          "VolumeSize": 50,
          "VolumeType": "gp2"
        }
      },
      {
        "DeviceName": "xvdf",
        "Ebs": {
          "DeleteOnTermination": true,
          "VolumeSize": 100,
          "VolumeType": "gp2"
        }
      }
    ],
    "IamInstanceProfile": {
      "Arn": "{{ createSSMInstanceProfile.instanceProfileArn }}"
    },
    "ImageId": "{{ getLatestWindowsAmi.amiId }}",
    "InstanceType": "t2.micro",
    "MaxCount": 1,
    "MinCount": 1,
    "PrivateIpAddress": "10.0.100.50",
    "SecurityGroupIds": [
      "{{ createDCSecGroup.dcSecGroupId }}"
    ],
    "SubnetId": "{{ createDCSubnet1.firstSubnetId }}",
    "TagSpecifications": [
      {"Key": "Name", "Value": "new-instance"}]
  }
}
"ResourceType": "instance",
"Tags": [
  {
    "Key": "Name",
    "Value": "SampleDC1"
  }
],
"outputs": [
  {
    "Name": "pdcInstanceId",
    "Selector": ".Instances[0].InstanceId",
    "Type": "String"
  }
],
"nextStep": "launchDC2"
},
{
  "name": "launchDC2",
  "action": "aws:executeAwsApi",
  "onFailure": "Abort",
  "inputs": {
    "Service": "ec2",
    "Api": "RunInstances",
    "BlockDeviceMappings": [
      {
        "DeviceName": "/dev/sdal",
        "Ebs": {
          "DeleteOnTermination": true,
          "VolumeSize": 50,
          "VolumeType": "gp2"
        }
      },
      {
        "DeviceName": "xvdf",
        "Ebs": {
          "DeleteOnTermination": true,
          "VolumeSize": 100,
          "VolumeType": "gp2"
        }
      }
    ],
    "IamInstanceProfile": {
      "Arn": "{{ createSSMInstanceProfile.instanceProfileArn }}"
    },
    "ImageId": "{{ getLatestWindowsAmi.amiId }}",
    "InstanceType": "t2.micro",
    "MaxCount": 1,
    "MinCount": 1,
    "PrivateIpAddress": "10.0.101.50",
    "SecurityGroupIds": ["{{ createDCSecGroup.dcSecGroupId }}"],
    "SubnetId": "{{ createDCSubnet2.secondSubnetId }}",
    "TagSpecifications": [
      {
        "ResourceType": "instance",
        "Tags": [
          {
            "Key": "Name",
            "Value": "SampleDC2"
          }
        ]
      }
    ]
  }
},
  "outputs": [
    {
      "Name": "adcInstanceId",
      "Selector": "$.Instances[0].InstanceId",
      "Type": "String"
    }
  ],
  "nextStep": "verifyDCInstanceState"
],
  "name": "verifyDCInstanceState",
  "action": "aws:waitForAwsResourceProperty",
  "inputs": {
    "Service": "ec2",
    "Api": "DescribeInstanceStatus",
    "IncludeAllInstances": true,
    "InstanceIds": [
      "{{ launchDC1.pdcInstanceId }}",
      "{{ launchDC2.adcInstanceId }}"
    ],
    "PropertySelector": "$.InstanceStatuses[0].InstanceState.Name",
    "DesiredValues": ["running"]
  },
  "nextStep": "verifyInstancesOnlineSSM"
],
  "name": "verifyInstancesOnlineSSM",
  "action": "aws:waitForAwsResourceProperty",
  "timeoutSeconds": 600,
  "inputs": {
    "Service": "ssm",
    "Api": "DescribeInstanceInformation",
    "InstanceInformationFilterList": [
      {
        "key": "InstanceIds",
        "valueSet": [
          "{{ launchDC1.pdcInstanceId }}",
          "{{ launchDC2.adcInstanceId }}"
        ]
      }
    ],
    "PropertySelector": "$.InstanceInformationList[0].PingStatus",
    "DesiredValues": ["Online"]
  },
  "nextStep": "installADRoles"
],
  "name": "installADRoles",
  "action": "aws:runCommand",
  "inputs": {
    "DocumentName": "AWS-RunPowerShellScript",
    "InstanceIds": [
      "{{ launchDC1.pdcInstanceId }}",
      "{{ launchDC2.adcInstanceId }}"
    ],
    "Parameters": {
      "commands": [
        "try {
          Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools",
        "]
      }
    }
  },
"catch {",
  "Write-Error \"Failed to install ADDS Role.\"",
"
"
"
"nextStep": "setAdminPassword"
},

"name": "setAdminPassword",
"action": "aws:runCommand",
"inputs": {
  "DocumentName": "AWS-RunPowerShellScript",
  "InstanceIds": [
    {{ launchDC1.pdcInstanceId }}
  ],
  "Parameters": {
    "commands": [
      "net user Administrator \"sampleAdminPass123!\"
    ]
  }
},

"nextStep": "createForest"
},

"name": "createForest",
"action": "aws:runCommand",
"inputs": {
  "DocumentName": "AWS-RunPowerShellScript",
  "InstanceIds": [
    {{ launchDC1.pdcInstanceId }}
  ],
  "Parameters": {
    "commands": [
      "$dsrmPass = 'sample123!' | ConvertTo-SecureString -asPlainText -Force",
      "try {
        Install-ADDSForest -DomainName \"sample.com\" -DomainMode 6 -ForestMode 6 -InstallDNS -DatabasePath \"D:\NTDS\" -SysvolPath \"D:\SYSVOL\" -SafeModeAdministratorPassword $dsrmPass -Force",
      "try {
        Add-DnsServerForwarder -IPAddress \"10.0.100.2\"
      }"
    ]
  }
},

"nextStep": "associateDhcpOptions"
},

"name": "associateDhcpOptions",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
  "Service": "ec2",
  "Api": "AssociateDhcpOptions",
  "DhcpOptionsId": "{{ createDhcpOptions.dhcpOptionsId }}",
  "VpcId": "{{ createVpc.vpcId }}"
},

"nextStep": "waitForADServices"
Restore a root volume from the latest snapshot

The operating system on a root volume can become corrupted for various reasons. For example, following a patching operation, instances may fail to boot successfully due to a corrupted kernel or registry. Automating common troubleshooting tasks, like restoring a root volume from the latest snapshot taken before the patching operation, can reduce downtime and expedite your troubleshooting efforts. AWS Systems Manager Automation actions can help you accomplish this.

The following sample AWS Systems Manager Automation document performs these actions.

- Uses the **aws:executeAwsApi** Automation action to retrieve details from the root volume of the instance.
- Uses the **aws:executeScript** Automation action to retrieve the latest snapshot for the root volume.
- Uses the **aws:branch** Automation action to continue the execution if a snapshot is found for the root volume.

**YAML**
---
description: Custom Automation Troubleshooting Sample
schemaVersion: '0.3'
assumeRole: "{{ AutomationAssumeRole }}"
parameters:
  AutomationAssumeRole:
    type: String
    description: "(Required) The ARN of the role that allows Automation to perform
    the actions on your behalf. If no role is specified, Systems Manager Automation
    uses your IAM permissions to execute this document."
    default: ''
  InstanceId:
    type: String
    description: "(Required) The Instance Id whose root EBS volume you want to
    restore the latest Snapshot."
    default: ''
mainSteps:
- name: getInstanceDetails
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeInstances
  InstanceIds:
    - "{{ InstanceId }}"
  outputs:
    - Name: availabilityZone
      Selector: "$\text{.Reservations[0].Instances[0].Placement.AvailabilityZone}"
      Type: String
    - Name: rootDeviceName
      Selector: "$\text{.Reservations[0].Instances[0].RootDeviceName}"
      Type: String
  nextStep: getRootVolumeId
- name: getRootVolumeId
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DescribeVolumes
  Filters:
    - Name: attachment.device
      Values: ["{{ getInstanceDetails.rootDeviceName }}"]
    - Name: attachment.instance-id
      Values: ["{{ InstanceId }}"]
  outputs:
    - Name: rootVolumeId
      Selector: "$\text{.Volumes[0].VolumeId}"
      Type: String
  nextStep: getSnapshotsByStartTime
- name: getSnapshotsByStartTime
  action: aws:executeScript
  onFailure: Abort
  inputs:
    Runtime: python3.6
    Handler: getSnapshotsByStartTime
  timeoutSeconds: 45
  InputPayload:
    rootVolumeId: "{{ getRootVolumeId.rootVolumeId }}"
  Script:
    def getSnapshotsByStartTime(events,context):
      import boto3
      #Initialize client
      ec2 = boto3.client('ec2')
      rootVolumeId = events['rootVolumeId']
      snapshotsQuery = ec2.describe_snapshots(
---
Filters=[
  {
    "Name": "volume-id",
    "Values": [rootVolumeId]
  }
]

if not snapshotsQuery['Snapshots']:
  noSnapshotFoundString = "NoSnapshotFound"
  return { 'noSnapshotFound': noSnapshotFoundString }
else:
  jsonSnapshots = snapshotsQuery['Snapshots']
  sortedSnapshots = sorted(jsonSnapshots, key=lambda k: k['StartTime'], reverse=True)
  latestSortedSnapshotId = sortedSnapshots[0]['SnapshotId']
  return { 'latestSnapshotId': latestSortedSnapshotId }

outputs:
- Name: Payload
  Selector: $.Payload
  Type: StringMap
- Name: latestSnapshotId
  Selector: $.Payload.latestSnapshotId
  Type: String
- Name: noSnapshotFound
  Selector: $.Payload.noSnapshotFound
  Type: String
nextStep: branchFromResults
- name: branchFromResults
  action: aws:branch
  onFailure: Abort
  inputs:
    Choices:
    - NextStep: createNewRootVolumeFromSnapshot
    Not:
      Variable: "{{ getSnapshotsByStartTime.noSnapshotFound }}"
      StringEquals: "NoSnapshotFound"
      isEnd: true
- name: createNewRootVolumeFromSnapshot
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateVolume
    AvailabilityZone: "{{ getInstanceDetails.availabilityZone }}"
    SnapshotId: "{{ getSnapshotsByStartTime.latestSnapshotId }}"
  outputs:
    - Name: newRootVolumeId
      Selector: "$VolumeId"
      Type: String
nextStep: stopInstance
- name: stopInstance
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: StopInstances
    InstanceIds:
      - "{{ InstanceId }}"
nextStep: verifyVolumeAvailability
- name: verifyVolumeAvailability
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 120
  inputs:
    Service: ec2
    Api: DescribeVolumes
    VolumeIds:
- "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
  PropertySelector: "$.Volumes[0].State"
  DesiredValues:
  - "available"
  nextStep: verifyInstanceStopped
- name: verifyInstanceStopped
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 120
  inputs:
    Service: ec2
    Api: DescribeInstances
    InstanceIds:
    - "{{ InstanceId }}"
    PropertySelector: "$.Reservations[0].Instances[0].State.Name"
    DesiredValues:
    - "stopped"
  nextStep: detachRootVolume
- name: detachRootVolume
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: DetachVolume
    VolumeId: "{{ getRootVolumeId.rootVolumeId }}"
  nextStep: verifyRootVolumeDetached
- name: verifyRootVolumeDetached
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 30
  inputs:
    Service: ec2
    Api: DescribeVolumes
    VolumeIds:
    - "{{ getRootVolumeId.rootVolumeId }}"
    PropertySelector: "$.Volumes[0].State"
    DesiredValues:
    - "available"
  nextStep: attachNewRootVolume
- name: attachNewRootVolume
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: AttachVolume
    Device: "{{ getInstanceDetails.rootDeviceName }}"
    InstanceId: "{{ InstanceId }}"
    VolumeId: "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
  nextStep: verifyNewRootVolumeAttached
- name: verifyNewRootVolumeAttached
  action: aws:waitForAwsResourceProperty
  timeoutSeconds: 30
  inputs:
    Service: ec2
    Api: DescribeVolumes
    VolumeIds:
    - "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
    PropertySelector: "$.Volumes[0].Attachments[0].State"
    DesiredValues:
    - "attached"
  nextStep: startInstance
- name: startInstance
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: StartInstances
    InstanceIds:
json

{
  "description": "Custom Automation Troubleshooting Sample",
  "schemaVersion": "0.3",
  "assumeRole": "{{ AutomationAssumeRole }}",
  "parameters": {
    "AutomationAssumeRole": {
      "type": "String",
      "description": "(Required) The ARN of the role that allows Automation
to perform the actions on your behalf. If no role is specified, Systems Manager
Automation uses your IAM permissions to execute this document.",
      "default": ""
    },
    "InstanceId": {
      "type": "String",
      "description": "(Required) The Instance Id whose root EBS volume you want to
restore the latest Snapshot.",
      "default": ""
    }
  },
  "mainSteps": [
    {
      "name": "getInstanceDetails",
      "action": "aws:executeAwsApi",
      "onFailure": "Abort",
      "inputs": {
        "Service": "ec2",
        "Api": "DescribeInstances",
        "InstanceIds": [
          "{{ InstanceId }}"
        ]
      },
      "outputs": [
        {
          "Name": "availabilityZone",
          "Selector": "$.Reservations[0].Instances[0].Placement.AvailabilityZone",
          "Type": "String"
        },
        {
          "Name": "rootDeviceName",
          "Selector": "$.Reservations[0].Instances[0].RootDeviceName",
          "Type": "String"
        }
      ],
      "nextStep": "getRootVolumeId"
    },
    {
      "name": "getRootVolumeId",
      "action": "aws:executeAwsApi",
      "onFailure": "Abort",
      "inputs": {
        "Service": "ec2",
        "Api": "DescribeVolumes",
        "Filters": [
          {
            "Name": "attachment.device",
            "Values": [
              "{{ getInstanceDetails.rootDeviceName }}"
            ]
          }
        ]
      }
    }
  ]
}
"Name": "attachment.instance-id",
"Values": [
  "{{ InstanceId }}"
]
},
"outputs": [
  {
    "Name": "rootVolumeId",
    "Selector": ".Volumes[0].VolumeId",
    "Type": "String"
  }
],
"nextStep": "getSnapshotsByStartTime"
},
{
  "name": "getSnapshotsByStartTime",
  "action": "aws:executeScript",
  "timeoutSeconds": 45,
  "onFailure": "Continue",
  "inputs": {
    "Runtime": "python3.6",
    "Handler": "getSnapshotsByStartTime",
    "InputPayload": {
      "rootVolumeId": "{{ getRootVolumeId.rootVolumeId }}"
    },
    "Attachment": "getSnapshotsByStartTime.py"
  },
  "outputs": [
    {
      "Name": "Payload",
      "Selector": "$.Payload",
      "Type": "StringMap"
    },
    {
      "Name": "latestSnapshotId",
      "Selector": "$.Payload.latestSnapshotId",
      "Type": "String"
    },
    {
      "Name": "noSnapshotFound",
      "Selector": "$.Payload.noSnapshotFound",
      "Type": "String"
    }
  ],
  "nextStep": "branchFromResults"
},
{
  "name": "branchFromResults",
  "action": "aws:branch",
  "onFailure": "Abort",
  "inputs": {
    "Choices": [
      {
        "NextStep": "createNewRootVolumeFromSnapshot",
        "Not": {
          "Variable": "{{ getSnapshotsByStartTime.noSnapshotFound }}",
          "StringEquals": "NoSnapshotFound"
        }
      }
    ],
    "isEnd": true
  },
"name": "createNewRootVolumeFromSnapshot",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
    "Service": "ec2",
    "Api": "CreateVolume",
    "AvailabilityZone": "{{ getInstanceDetails.availabilityZone }}",
    "SnapshotId": "{{ getSnapshotsByStartTime.latestSnapshotId }}"
},
"outputs": [
    {
        "Name": "newRootVolumeId",
        "Selector": "$\.VolumeId",
        "Type": "String"
    }
],
"nextStep": "stopInstance"
},
{
    "name": "stopInstance",
    "action": "aws:executeAwsApi",
    "onFailure": "Abort",
    "inputs": {
        "Service": "ec2",
        "Api": "StopInstances",
        "InstanceIds": [
            "{{ InstanceId }}"
        ]
    },
    "nextStep": "verifyVolumeAvailability"
},
{
    "name": "verifyVolumeAvailability",
    "action": "aws:waitForAwsResourceProperty",
    "timeoutSeconds": 120,
    "inputs": {
        "Service": "ec2",
        "Api": "DescribeVolumes",
        "VolumeIds": [
            "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
        ],
        "PropertySelector": "$\.Volumes[0].State",
        "DesiredValues": [
            "available"
        ]
    },
    "nextStep": "verifyInstanceStopped"
},
{
    "name": "verifyInstanceStopped",
    "action": "aws:waitForAwsResourceProperty",
    "timeoutSeconds": 120,
    "inputs": {
        "Service": "ec2",
        "Api": "DescribeInstances",
        "InstanceIds": [
            "{{ InstanceId }}"
        ],
        "PropertySelector": "$\.Reservations[0].Instances[0].State.Name",
        "DesiredValues": [
            "stopped"
        ]
    },
    "nextStep": "detachRootVolume"
},

"name": "detachRootVolume",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
    "Service": "ec2",
    "Api": "DetachVolume",
    "VolumeId": "{{ getRootVolumeId.rootVolumeId }}"
},
"nextStep": "verifyRootVolumeDetached"
},

"name": "verifyRootVolumeDetached",
"action": "aws:waitForAwsResourceProperty",
"timeoutSeconds": 30,
"inputs": {
    "Service": "ec2",
    "Api": "DescribeVolumes",
    "VolumeIds": [
        "{{ getRootVolumeId.rootVolumeId }}"
    ],
    "PropertySelector": ".Volumes[0].State",
    "DesiredValues": [
        "available"
    ]
},
"nextStep": "attachNewRootVolume"
},

"name": "attachNewRootVolume",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
    "Service": "ec2",
    "Api": "AttachVolume",
    "Device": "{{ getInstanceDetails.rootDeviceName }}",
    "InstanceId": "{{ InstanceId }}",
    "VolumeId": "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
},
"nextStep": "verifyNewRootVolumeAttached"
},

"name": "verifyNewRootVolumeAttached",
"action": "aws:waitForAwsResourceProperty",
"timeoutSeconds": 30,
"inputs": {
    "Service": "ec2",
    "Api": "DescribeVolumes",
    "VolumeIds": [
        "{{ createNewRootVolumeFromSnapshot.newRootVolumeId }}"
    ],
    "PropertySelector": ".Volumes[0].Attachments[0].State",
    "DesiredValues": [
        "attached"
    ]
},
"nextStep": "startInstance"
},

"name": "startInstance",
"action": "aws:executeAwsApi",
"onFailure": "Abort",
"inputs": {
    "Service": "ec2",
    "Api": "StartInstances",
    "InstanceIds": [
        "{{ InstanceId }}"
    ]
}
Create an AMI and cross-Region copy

Creating an Amazon Machine Image (AMI) of an instance is a common process used in backup and recovery. You might also choose to copy an AMI to another Region as part of a disaster recovery architecture. Automating common maintenance tasks can reduce downtime if an issue requires failover. AWS Systems Manager Automation actions can help you accomplish this.

The following sample AWS Systems Manager Automation document below performs these actions.

- Uses the `aws:executeAwsApi` Automation action to create an AMI.
- Uses the `aws:waitForAwsResourceProperty` Automation action to confirm the availability of the AMI.
- Uses the `aws:executeScript` Automation action to copy the AMI to the destination Region.

YAML

```yaml
---
description: Custom Automation Backup and Recovery Sample
schemaVersion: '0.3'
asumeRole: "{{ AutomationAssumeRole }}"
parameters:
  AutomationAssumeRole:
    type: String
    description: "(Required) The ARN of the role that allows Automation to perform the actions on your behalf. If no role is specified, Systems Manager Automation uses your IAM permissions to execute this document."
    default: ''
  InstanceId:
    type: String
    description: "(Required) The ID of the EC2 instance."
    default: ''
mainSteps:
- name: createImage
  action: aws:executeAwsApi
  onFailure: Abort
  inputs:
    Service: ec2
    Api: CreateImage
    InstanceId: "{{ InstanceId }}"
    Name: "Automation Image for {{ InstanceId }}"
    NoReboot: false
  outputs:
    - Name: newImageId
      Selector: ".ImageId"
      Type: String
      nextStep: verifyImageAvailability
- name: verifyImageAvailability
```
action: aws:waitForAwsResourceProperty
timeoutSeconds: 600
inputs:
  Service: ec2
  Api: DescribeImages
  ImageIds:
  - "{{ createImage.newImageId }}"
  PropertySelector: "$.Images[0].State"
  DesiredValues:
  - available
nextStep: copyImage
  name: copyImage
action: aws:executeScript
timeoutSeconds: 45
onFailure: Abort
inputs:
  Runtime: python3.6
  Handler: crossRegionImageCopy
  InputPayload:
    newImageId : "{{ createImage.newImageId }}"
  Script: |
    def crossRegionImageCopy(events,context):
      import boto3
      #Initialize client
      ec2 = boto3.client('ec2', region_name='us-east-1')
      newImageId = events['newImageId']
      ec2.copy_image(
          Name='DR Copy for ' + newImageId,
          SourceImageId=newImageId,
          SourceRegion='us-west-2'
      )
}

JSON

{
  "description": "Custom Automation Backup and Recovery Sample",
  "schemaVersion": "0.3",
  "assumeRole": "{{ AutomationAssumeRole }}",
  "parameters": {
    "AutomationAssumeRole": {
      "type": "String",
      "description": "(Required) The ARN of the role that allows Automation to perform\nthe actions on your behalf. If no role is specified, Systems Manager Automation\nuses your IAM permissions to execute this document.",
      "default": ""
    },
    "InstanceId": {
      "type": "String",
      "description": "(Required) The ID of the EC2 instance.",
      "default": ""
    }
  },
  "mainSteps": [
    {
      "name": "createImage",
      "action": "aws:executeAwsApi",
      "onFailure": "Abort",
      "inputs": {
        "Service": "ec2",
        "Api": "CreateImage",
        "InstanceId": "{{ InstanceId }}",
        "Name": "Automation Image for {{ InstanceId }}",
      },
    }
  ]
}
"NoReboot": false,
"outputs": [
{
 "Name": "newImageId",
 "Selector": "$.ImageId",
 "Type": "String"
}],
"nextStep": "verifyImageAvailability"
},
{
 "name": "verifyImageAvailability",
 "action": "aws:waitForAwsResourceProperty",
 "timeoutSeconds": 600,
 "inputs": {
 "Service": "ec2",
 "Api": "DescribeImages",
 "ImageIds": [
 "{{ createImage.newImageId }}"
 ],
 "PropertySelector": "$.Images[0].State",
 "DesiredValues": [
 "available"
 ]
 },
 "nextStep": "copyImage"
},
{
 "name": "copyImage",
 "action": "aws:executeScript",
 "timeoutSeconds": 45,
 "onFailure": "Abort",
 "inputs": {
 "Runtime": "python3.6",
 "Handler": "crossRegionImageCopy",
 "InputPayload": {
 "newImageId": "{{ createImage.newImageId }}"
 },
 "Attachment": "crossRegionImageCopy.py"
 }
],
 "files": {
 "crossRegionImageCopy.py": {
 "checksums": {
 "sha256": "sampleETagValue"
 }
 }
 }
}

**Automation walkthroughs**

The following walkthroughs help you get started with Systems Manager Automation using predefined Automation documents, or playbooks.

Before you begin, you must configure Automation roles and permissions. For more information, see Getting started with Automation (p. 281). For information about creating a custom Automation document, see Walkthrough: Patch a Windows Server AMI (p. 576).

**Walkthroughs**
• Patching Amazon Machine Images (p. 569)
• Using AWS support self-service Automations (p. 591)
• Walkthrough: Using input transformers with Automation (p. 601)
• Walkthrough: Using Automation with Jenkins (p. 601)
• Walkthrough: Using Document Builder to create a custom Automation document (p. 604)

Patching Amazon Machine Images

This section includes walkthroughs that describe how to patch or update Amazon Machine Images (AMIs).

Topics

• Walkthrough: Patch a Linux AMI (console) (p. 569)
• Walkthrough: Patch a Linux AMI (AWS CLI) (p. 572)
• Walkthrough: Patch a Windows Server AMI (p. 576)
• Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store (p. 580)
• Walkthrough: Patch an AMI and update an Auto Scaling group (p. 586)

Walkthrough: Patch a Linux AMI (console)

This Systems Manager Automation walkthrough shows you how to use the console and the Systems Manager AWS-UpdateLinuxAmi document to automatically patch a Linux AMI with the latest versions of packages that you specify. The AWS-UpdateLinuxAmi document also automates the installation of additional site-specific packages and configurations. You can update a variety of Linux distributions using this walkthrough, including Ubuntu, CentOS, RHEL, SLES, or Amazon Linux AMIs. For a full list of supported Linux versions, see Patch Manager prerequisites (p. 912).

The AWS-UpdateLinuxAmi document enables you to automate image-maintenance tasks without having to author the workflow in JSON or YAML. You can use the AWS-UpdateLinuxAmi document to perform the following types of tasks.

• Upgrade all distribution packages and Amazon software on an Amazon Linux, Red Hat, Ubuntu, SLES, or CentOS Amazon Machine Image (AMI). This is the default document behavior.
• Install SSM Agent on an existing image to enable Systems Manager capabilities, such as remote command execution using Run Command or software inventory collection using Inventory.
• Install additional software packages.

Before You Begin

Before you begin working with Automation documents, configure roles and, optionally, CloudWatch Events for Automation. For more information, see Getting started with Automation (p. 281). This walkthrough also requires that you specify the name of an AWS Identity and Access Management (IAM) instance profile. For more information about creating an IAM instance profile, see Create an IAM instance profile for Systems Manager (p. 30).

The AWS-UpdateLinuxAmi document accepts the following input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceAmiId</td>
<td>String</td>
<td>(Required) The source AMI ID.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>iamInstanceProfileName</td>
<td>String</td>
<td>(Required) The name of the IAM instance profile role you created in Create an IAM instance profile for Systems Manager (p. 30). The instance profile role gives Automation permission to perform actions on your instances, such as running commands or starting and stopping services. The Automation document uses only the name of the instance profile role. If you specify the Amazon Resource Name (ARN), the Automation execution fails.</td>
</tr>
<tr>
<td>AutomationAssumeRole</td>
<td>String</td>
<td>(Required) The name of the IAM service role you created in Getting started with Automation (p. 281). The service role (also called an assume role) gives Automation permission to assume your IAM role and perform actions on your behalf. For example, the service role allows Automation to create a new AMI when running the aws:createImage action in an Automation document. For this parameter, the complete ARN must be specified.</td>
</tr>
<tr>
<td>TargetAmiName</td>
<td>String</td>
<td>(Optional) The name of the new AMI after it is created. The default name is a system-generated string that includes the source AMI ID, and the creation time and date.</td>
</tr>
<tr>
<td>InstanceType</td>
<td>String</td>
<td>(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is t2.micro.</td>
</tr>
<tr>
<td>PreUpdateScript</td>
<td>String</td>
<td>(Optional) URL of a script to run before updates are applied. Default (&quot;none&quot;) is to not run a script.</td>
</tr>
<tr>
<td>PostUpdateScript</td>
<td>String</td>
<td>(Optional) URL of a script to run after package updates are applied. Default (&quot;none&quot;) is to not run a script.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IncludePackages</td>
<td>String</td>
<td>(Optional) Only update these named packages. By default (&quot;all&quot;), all available updates are applied.</td>
</tr>
<tr>
<td>ExcludePackages</td>
<td>String</td>
<td>(Optional) Names of packages to hold back from updates, under all conditions. By default (&quot;none&quot;), no package is excluded.</td>
</tr>
</tbody>
</table>

### Automation Steps

The **AWS-UpdateLinuxAmi** document includes the following Automation steps, by default.

**Step 1: launchInstance (aws:runInstances action)**

This step launches an instance using Amazon EC2 userdata and an IAM instance profile role. Userdata installs the appropriate SSM Agent, based on the operating system. Installing SSM Agent enables you to utilize Systems Manager capabilities such as Run Command, State Manager, and Inventory.

**Step 2: updateOSSoftware (aws:runCommand action)**

This step runs the following commands on the launched instance:

- Downloads an update script from Amazon S3.
- Runs an optional pre-update script.
- Updates distribution packages and Amazon software.
- Runs an optional post-update script.

The execution log is stored in the /tmp folder for the user to view later.

If you want to upgrade a specific set of packages, you can supply the list using the `IncludePackages` parameter. When provided, the system attempts to update only these packages and their dependencies. No other updates are performed. By default, when no `include` packages are specified, the program updates all available packages.

If you want to exclude upgrading a specific set of packages, you can supply the list to the `ExcludePackages` parameter. If provided, these packages remain at their current version, independent of any other options specified. By default, when no `exclude` packages are specified, no packages are excluded.

**Step 3: stopInstance (aws:changeInstanceState action)**

This step stops the updated instance.

**Step 4: createImage (aws:createImage action)**

This step creates a new AMI with a descriptive name that links it to the source ID and creation time. For example: “AMI Generated by EC2 Automation on {{global:DATE_TIME}} from {{SourceAmiId}}” where DATE_TIME and SourceID represent Automation variables.

**Step 5: terminateInstance (aws:changeInstanceState action)**

This step cleans up the execution by terminating the running instance.

**Output**

The execution returns the new AMI ID as output.
Note
By default, when Automation runs the AWS-UpdateLinuxAmi document, the system creates a temporary instance in the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:
VPC not defined 400
To solve this problem, you must make a copy of the AWS-UpdateLinuxAmi document and specify a subnet ID. For more information, see VPC not defined 400 (p. 610).

To create a patched AMI using Automation (AWS Systems Manager)
2. In the navigation pane, choose Automation.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Automation.
3. Choose Execute automation.
4. In the Automation document list, choose AWS-UpdateLinuxAmi.
5. In the Document details section, verify that Document version is set to Default version at runtime.
6. Choose Next.
7. In the Execution mode section, choose Simple Execution.
8. In the Input parameters section, enter the information you collected in the Before You Begin section.
9. Choose Execute. The console displays the status of the Automation execution.

After the workflow finishes, launch a test instance from the updated AMI to verify changes.

Note
If any step in the workflow fails, information about the failure is listed on the Automation Executions page. The workflow is designed to terminate the temporary instance after successfully completing all tasks. If a step fails, the system might not terminate the instance. So if a step fails, manually terminate the temporary instance.

Walkthrough: Patch a Linux AMI (AWS CLI)
This Systems Manager Automation walkthrough shows you how to use the AWS CLI and the Systems Manager AWS-UpdateLinuxAmi document to automatically patch a Linux AMI with the latest versions of packages that you specify. The AWS-UpdateLinuxAmi document also automates the installation of additional site-specific packages and configurations. You can update a variety of Linux distributions using this walkthrough, including Ubuntu, CentOS, RHEL, SLES, or Amazon Linux AMIs. For a full list of supported Linux versions, see Patch Manager prerequisites (p. 912).

The AWS-UpdateLinuxAmi document enables you to automate image-maintenance tasks without having to author the workflow in JSON or YAML. You can use the AWS-UpdateLinuxAmi document to perform the following types of tasks.

- Upgrade all distribution packages and Amazon software on an Amazon Linux, Red Hat, Ubuntu, SLES, or CentOS Amazon Machine Image (AMI). This is the default document behavior.
- Install SSM Agent on an existing image to enable Systems Manager capabilities, such as remote command execution using Run Command or software inventory collection using Inventory.
- Install additional software packages.

Before You Begin
Before you begin working with Automation documents, configure roles and, optionally, CloudWatch Events for Automation. For more information, see Getting started with Automation (p. 281). This walkthrough also requires that you specify the name of an AWS Identity and Access Management (IAM) instance profile. For more information about creating an IAM instance profile, see Create an IAM instance profile for Systems Manager (p. 30).

The AWS-UpdateLinuxAmi document accepts the following input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceAmiId</td>
<td>String</td>
<td>(Required) The source AMI ID. You can automatically reference the latest ID of an EC2 AMI for Linux by using a Systems Manager Parameter Store public parameter. For more information, see Query for the latest Amazon Linux AMI IDs using AWS Systems Manager Parameter Store.</td>
</tr>
<tr>
<td>IamInstanceProfileName</td>
<td>String</td>
<td>(Required) The name of the IAM instance profile role you created in Create an IAM instance profile for Systems Manager (p. 30). The instance profile role gives Automation permission to perform actions on your instances, such as running commands or starting and stopping services. The Automation document uses only the name of the instance profile role. If you specify the Amazon Resource Name (ARN), the Automation execution fails.</td>
</tr>
<tr>
<td>AutomationAssumeRole</td>
<td>String</td>
<td>(Required) The name of the IAM service role you created in Getting started with Automation (p. 281). The service role (also called an assume role) gives Automation permission to assume your IAM role and perform actions on your behalf. For example, the service role allows Automation to create a new AMI when running the aws:createImage action in an Automation document. For this parameter, the complete ARN must be specified.</td>
</tr>
</tbody>
</table>
| TargetAmiName           | String  | (Optional) The name of the new AMI after it is created. The default name is a system-generated string that includes }
### Parameter Types and Descriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>the source AMI ID, and the creation time and date.</td>
</tr>
<tr>
<td>InstanceType</td>
<td>String</td>
<td>(Optional) The type of instance to launch as the workspace host. Instance types vary by region.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default type is t2.micro.</td>
</tr>
<tr>
<td>PreUpdateScript</td>
<td>String</td>
<td>(Optional) URL of a script to run before updates are applied. Default (&quot;none&quot;) is to not run a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>script.</td>
</tr>
<tr>
<td>PostUpdateScript</td>
<td>String</td>
<td>(Optional) URL of a script to run after package updates are applied. Default (&quot;none&quot;) is to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not run a script.</td>
</tr>
<tr>
<td>IncludePackages</td>
<td>String</td>
<td>(Optional) Only update these named packages. By default (&quot;all&quot;), all available updates are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>applied.</td>
</tr>
<tr>
<td>ExcludePackages</td>
<td>String</td>
<td>(Optional) Names of packages to hold back from updates, under all conditions. By default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(&quot;none&quot;), no package is excluded.</td>
</tr>
</tbody>
</table>

### Automation Steps

The AWS-UpdateLinuxAmi document includes the following Automation steps, by default.

**Step 1: launchInstance (aws:runInstances action)**

This step launches an instance using Amazon EC2 userdata and an IAM instance profile role. Userdata installs the appropriate SSM Agent, based on the operating system. Installing SSM Agent enables you to utilize Systems Manager capabilities such as Run Command, State Manager, and Inventory.

**Step 2: updateOSSoftware (aws:runCommand action)**

This step runs the following commands on the launched instance:

- Downloads an update script from Amazon S3.
- Runs an optional pre-update script.
- Updates distribution packages and Amazon software.
- Runs an optional post-update script.

The execution log is stored in the /tmp folder for the user to view later.

If you want to upgrade a specific set of packages, you can supply the list using the `IncludePackages` parameter. When provided, the system attempts to update only these packages and their dependencies. No other updates are performed. By default, when no include packages are specified, the program updates all available packages.

If you want to exclude upgrading a specific set of packages, you can supply the list to the `ExcludePackages` parameter. If provided, these packages remain at their current version.
independent of any other options specified. By default, when no exclude packages are specified, no packages are excluded.

**Step 3: stopInstance (aws:changeInstanceState action)**

This step stops the updated instance.

**Step 4: createImage (aws:createImage action)**

This step creates a new AMI with a descriptive name that links it to the source ID and creation time. For example: "AMI Generated by EC2 Automation on {{global:DATE_TIME}} from {{SourceAmiId}}" where DATE_TIME and SourceID represent Automation variables.

**Step 5: terminateInstance (aws:changeInstanceState action)**

This step cleans up the execution by terminating the running instance.

**Output**

The execution returns the new AMI ID as output.

**Note**

By default, when Automation runs the AWS-UpdateLinuxAmi document, the system creates a temporary instance in the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:

VPC not defined 400

To solve this problem, you must make a copy of the AWS-UpdateLinuxAmi document and specify a subnet ID. For more information, see VPC not defined 400 (p. 610).

**To create a patched AMI using Automation**

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to run the AWS-UpdateLinuxAmi document and run the Automation workflow. In the parameters section, specify an AMI source ID, an EC2 instance profile role, and your Automation service role.

   ```bash
   aws ssm start-automation-execution \
   --document-name "AWS-UpdateLinuxAmi" \
   --parameters \ 
   SourceAmiId=ami-0080e4c5bc078760e, \
   IamInstanceProfileName=ManagedInstanceRole, \
   AutomationAssumeRole='arn:aws:iam::{{global:ACCOUNT_ID}}:role/AutomationServiceRole' 
   ```

   The command returns an execution ID. Copy this ID to the clipboard. You will use this ID to view the status of the workflow.

   ```json
   { 
       "AutomationExecutionId": "ID" 
   }
   ```

3. To view the workflow execution using the AWS CLI, run the following command:

   ```bash
   aws ssm describe-automation-executions 
   ```

4. To view details about the execution progress, run the following command.

   ```bash
   aws ssm get-automation-execution --automation-execution-id ID 
   ```
The update process can take 30 minutes or more to complete.

Note
You can also monitor the status of the workflow in the console. In the execution list, choose the execution you just ran and then choose the Steps tab. This tab shows you the status of the workflow actions.

After the workflow finishes, launch a test instance from the updated AMI to verify changes.

Note
If any step in the workflow fails, information about the failure is listed on the Automation Executions page. The workflow is designed to terminate the temporary instance after successfully completing all tasks. If a step fails, the system might not terminate the instance. So if a step fails, manually terminate the temporary instance.

Walkthrough: Patch a Windows Server AMI

The AWS-UpdateWindowsAmi document enables you to automate image maintenance tasks on your Amazon Windows AMIs without having to author the workflow in JSON or YAML. This document is supported for Windows Server 2008 R2 or later. You can use the AWS-UpdateWindowsAmi document to perform the following types of tasks.

- Install all Windows updates and upgrade Amazon software (default behavior).
- Install specific Windows updates and upgrade Amazon software.
- Customize an AMI using your scripts.

Before You Begin

Before you begin working with Automation documents, configure roles and, optionally, CloudWatch Events for Automation. For more information, see Getting started with Automation (p. 281). This walkthrough also requires that you specify the name of an AWS Identity and Access Management (IAM) instance profile. For more information about creating an IAM instance profile, see Create an IAM instance profile for Systems Manager (p. 30).

Note
Updates to SSM Agent are typically rolled out to different regions at different times. When you customize or update an AMI, use only source AMIs published for the region that you are working in. This will ensure that you are working with the latest SSM Agent released for that region and avoid compatibility issues.

The AWS-UpdateWindowsAmi document accepts the following input parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SourceAmiId</td>
<td>String</td>
<td>(Required) The source AMI ID. You can automatically reference the latest Windows Server AMI ID by using a Systems Manager Parameter Store public parameter. For more information, see Query for the latest Windows AMI IDs using AWS Systems Manager Parameter Store.</td>
</tr>
<tr>
<td>IamInstanceProfileName</td>
<td>String</td>
<td>(Required) The name of the IAM instance profile role you</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutomationAssumeRole</strong></td>
<td>String</td>
<td>(Required) The name of the IAM service role you created in <em>Getting started with Automation</em> (p. 281). The service role (also called an assume role) gives Automation permission to assume your IAM role and perform actions on your behalf. For example, the service role allows Automation to create a new AMI when running the <code>aws:createImage</code> action in an Automation document. For this parameter, the complete ARN must be specified.</td>
</tr>
<tr>
<td><strong>TargetAmiName</strong></td>
<td>String</td>
<td>(Optional) The name of the new AMI after it is created. The default name is a system-generated string that includes the source AMI ID, and the creation time and date.</td>
</tr>
<tr>
<td><strong>InstanceType</strong></td>
<td>String</td>
<td>(Optional) The type of instance to launch as the workspace host. Instance types vary by region. The default type is t2.medium.</td>
</tr>
<tr>
<td><strong>PreUpdateScript</strong></td>
<td>String</td>
<td>(Optional) A script to run before updating the AMI. Enter a script in the Automation document or at runtime as a parameter.</td>
</tr>
<tr>
<td><strong>PostUpdateScript</strong></td>
<td>String</td>
<td>(Optional) A script to run after updating the AMI. Enter a script in the Automation document or at runtime as a parameter.</td>
</tr>
</tbody>
</table>

- **Parameter**
- **Type**
- **Description**

created in [Create an IAM instance profile for Systems Manager](p. 30). The instance profile role gives Automation permission to perform actions on your instances, such as running commands or starting and stopping services. The Automation document uses only the name of the instance profile role. If you specify the Amazon Resource Name (ARN), the Automation execution fails.
### Parameter | Type | Description
--- | --- | ---
IncludeKbs | String | (Optional) Specify one or more Microsoft Knowledge Base (KB) article IDs to include. You can install multiple IDs using comma-separated values. Valid formats: KB9876543 or 9876543.

ExcludeKbs | String | (Optional) Specify one or more Microsoft Knowledge Base (KB) article IDs to exclude. You can exclude multiple IDs using comma-separated values. Valid formats: KB9876543 or 9876543.

Categories | String | (Optional) Specify one or more update categories. You can filter categories using comma-separated 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.

SeverityLevels | String | (Optional) Specify one or more MSRC severity levels associated with an update. You can filter severity levels using comma-separated 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.

---

**Automation Steps**

The `AWS-UpdateWindowsAmi` document includes the following Automation steps, by default.

**Step 1: launchInstance (aws:runInstances action)**

This step launches an instance with an IAM instance profile role from the specified `SourceAmiID`.

**Step 2: runPreUpdateScript (aws:runCommand action)**

This step enables you to specify a script as a string that runs before updates are installed.
Step 3: updateEC2Config (aws:runCommand action)

This step uses the AWS-InstallPowerShellModule public document to download an AWS public PowerShell module. Systems Manager verifies the integrity of the module by using an SHA-256 hash. Systems Manager then checks the operating system to determine whether to update EC2Config or EC2Launch. EC2Config runs on Windows Server 2008 R2 through Windows Server 2012 R2. EC2Launch runs on Windows Server 2016.

Step 4: updateSSMAgent (aws:runCommand action)

This step updates SSM Agent by using the AWS-UpdateSSMAgent public document.

Step 5: updateAWSPVDriver (aws:runCommand action)

This step updates AWS PV drivers by using the AWS-ConfigureAWSPackage public document.

Step 6: updateAwsEnaNetworkDriver (aws:runCommand action)

This step updates AWS ENA Network drivers by using the AWS-ConfigureAWSPackage public document.

Step 7: installWindowsUpdates (aws:runCommand action)

This step installs Windows updates by using the AWS-InstallWindowsUpdates public document. By default, Systems Manager searches for and installs all missing updates. You can change the default behavior by specifying one of the following parameters: IncludeKbs, ExcludeKbs, Categories, or SeverityLevels.

Step 8: runPostUpdateScript (aws:runCommand action)

This step enables you to specify a script as a string that runs after the updates have been installed.

Step 9: runSysprepGeneralize (aws:runCommand action)

This step uses the AWS-InstallPowerShellModule public document to download an AWS public PowerShell module. Systems Manager verifies the integrity of the module by using an SHA-256 hash. Systems Manager then runs sysprep using AWS-supported methods for either EC2Launch (Windows Server 2016) or EC2Config (Windows Server 2008 R2 through 2012 R2).

Step 10: stopInstance (aws:changeInstanceState action)

This step stops the updated instance.

Step 11: createImage (aws:createImage action)

This step creates a new AMI with a descriptive name that links it to the source ID and creation time. For example: "AMI Generated by EC2 Automation on {{global:DATE_TIME}} from {{SourceAmiId}}" where DATE_TIME and SourceID represent Automation variables.

Step 12: TerminateInstance (aws:changeInstanceState action)

This step cleans up the execution by terminating the running instance.

Output

This section enables you to designate the outputs of various steps or values of any parameter as the Automation output. By default, the output is the ID of the updated Windows AMI created by the execution.

Note

By default, when Automation runs the AWS-UpdateWindowsAmi document and creates a temporary instance, the system uses the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:

VPC not defined 400

To solve this problem, you must make a copy of the AWS-UpdateWindowsAmi document and specify a subnet ID. For more information, see VPC not defined 400 (p. 610).
To create a patched Windows AMI by using Automation

1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to run the AWS-UpdateWindowsAmi document. In the parameters section, specify an AMI source ID, an EC2 instance profile role, and your Automation service role. The example command below uses a recent Amazon EC2 AMI to minimize the number of patches that need to be applied. If you run this command more than once, you must specify a unique value for targetAMIname. AMI names must be unique.

   ```bash
   aws ssm start-automation-execution --document-name="AWS-UpdateWindowsAmi" --parameters
   SourceAmiId='ami-0246f4914689c475f',IamInstanceProfileName='ManagedInstanceProfile',AutomationAssumeRole='{{global:ACCOUNT_ID}}:role/AutomationServiceRole'
   
   The command returns an execution ID. Copy this ID to the clipboard. You will use this ID to view the status of the workflow.

   ```

   ```json
   {
     "AutomationExecutionId": "ID"
   }
   ```

3. To view the workflow execution using the AWS CLI, run the following command:

   ```bash
   aws ssm describe-automation-executions
   ```

4. To view details about the execution progress, run the following command.

   ```bash
   aws ssm get-automation-execution --automation-execution-id ID
   ```

**Note**
Depending on the number of patches applied, the Windows patching process run in this sample workflow can take 30 minutes or more to complete.

**Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store**

The following example expands on how to update a Windows AMI, as described in Walkthrough: Patch a Windows Server AMI (p. 576). This example uses the model where an organization maintains and periodically patches their own, proprietary AMIs rather than building from Amazon EC2 AMIs.

The following procedure shows how to automatically apply operating system (OS) patches to a Windows AMI that is already considered to be the most up-to-date or latest AMI. In the example, the default value of the parameter SourceAmiId is defined by a Systems Manager Parameter Store parameter called latestAmi. The value of latestAmi is updated by an AWS Lambda function invoked at the end of the Automation workflow. As a result of this Automation process, the time and effort spent patching AMIs is minimized because patching is always applied to the most up-to-date AMI.

**Before You Begin**

Configure Automation roles and, optionally, CloudWatch Events for Automation. For more information, see Getting started with Automation (p. 281).

**Contents**

- Task 1: Create a parameter in Systems Manager Parameter Store (p. 581)
• Task 2: Create an IAM role for AWS Lambda (p. 581)
• Task 3: Create an AWS Lambda function (p. 582)
• Task 4: Create an Automation document and patch the AMI (p. 584)

Task 1: Create a parameter in Systems Manager Parameter Store

Create a string parameter in Parameter Store that uses the following information:

• **Name:** latestAmi.
• **Value:** a Windows AMI ID. For example: ami-188d6e0e.

For information about how to create a Parameter Store string parameter, see Creating Systems Manager parameters (p. 249).

Task 2: Create an IAM role for AWS Lambda

Use the following procedure to create an IAM service role for AWS Lambda. This role includes the **AWSLambdaExecute** and **AmazonSSMFullAccess** managed policies. These policies give Lambda permission to update the value of the `latestAmi` parameter using a Lambda function and Systems Manager.

To create an IAM service role for Lambda

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**, and then choose **Create policy**.
3. Choose the **JSON** tab.
4. Replace the default content with the following. Be sure to replace `us-west-2` and `123456789012` with the Region and account you want to use. Replace `updateAmiFunction` with the name of your Lambda function.

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": "logs:CreateLogGroup",
       },
       {
         "Effect": "Allow",
         "Action": [
           "logs:CreateLogStream",
           "logs:PutLogEvents"
         ],
         "Resource": [
         ]
       }
     ]
   }
   
   5. Choose **Review policy**.
   6. On the **Review policy** page, for **Name**, enter a name for the inline policy, such as **amiLambda**.
   7. Choose **Create policy**.
8. Repeat steps 2 and 3.
9. Replace the default content with the following. Be sure to replace us-west-2 and 123456789012 with the Region and account you want to use.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:PutParameter",
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeParameters",
            "Resource": "*"
        }
    ]
}
```

11. On the Review policy page, for Name, enter a name for the inline policy, such as amiParameter.
12. Choose Create policy.
13. In the navigation pane, choose Roles, and then choose Create role.
14. Immediately under Choose the service that will use this role, choose Lambda, and then choose Next: Permissions.
15. On the Attach permissions policies page, use the Search field to locate the two policies you created earlier.
16. Select the check box next to the policies, and then choose Next: Tags.
17. (Optional) Add one or more tag key-value pairs to organize, track, or control access for this role, and then choose Next: Review.
18. For Role name, enter a name for your new role, such as lambda-ssm-role or another name that you prefer.

   **Note**
   Because various entities might reference the role, you cannot change the name of the role after it has been created.
19. Choose Create role.

**Task 3: Create an AWS Lambda function**

Use the following procedure to create a Lambda function that automatically updates the value of the latestAmi parameter.

**To create a Lambda function**

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. On the Create function page, choose Author from scratch.
4. For Function name, type Automation-UpdateSsmParam.
5. In the Runtime list, choose Python 3.8.
6. In the Permissions section, expand Choose or create an execution role.
7. Choose Use an existing role, and then choose the service role for Lambda that you created in Task 2.
8. Choose **Create function**.

9. In the **Function code** area, on the **lambda_function** tab, delete the pre-populated code in the field, and then paste the following code sample.

```python
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
        )

    responseString = 'Updated parameter %s with value %s.' % (event['parameterName'], event['parameterValue'])

    return responseString
```

10. Choose **Save**.

11. To test the Lambda function, from the **Select a test event** menu, choose **Configure test events**.

12. For **Event name**, enter a name for the test event, such as **MyTestEvent**.
13. Replace the existing text with the following JSON, replacing \texttt{your-ami-id} with the ID of the new
AMI to set as your \texttt{latestAmi} parameter value.

\begin{verbatim}
{
    "parameterName":"latestAmi",
    "parameterValue":"\texttt{your-ami-id}"
}
\end{verbatim}

14. Choose \textbf{Create}.

15. Choose \textbf{Test} to test the function. The output should state that the parameter was successfully
updated and include details about the update. For example, “Updated parameter latestAmi with value ami-123456”.

\section*{Task 4: Create an Automation document and patch the AMI}

Use the following procedure to create and run an Automation document that patches the AMI you
specified for the \texttt{latestAmi} parameter. After the Automation workflow completes, the value of \texttt{latestAmi}
is updated with the ID of the newly-patched AMI. Subsequent executions use the AMI created by the
previous execution.

\textbf{To create an Automation document and patch an AMI}

2. In the navigation pane, choose \textbf{Documents}.

\textbf{-or-}

If the AWS Systems Manager home page opens first, choose the menu icon (\textcircled{\textbf{3}}) to open the
navigation pane, and then choose \textbf{Documents} in the navigation pane.
3. Choose \textbf{Create automation}.
4. In the \textbf{Name} field, type \texttt{UpdateMyLatestWindowsAmi}.
5. Choose the \textbf{Editor} tab, and then choose \textbf{Edit}.
6. Replace the default contents in the \textbf{Document editor} field with following JSON sample document.

\textbf{Note}

You must change the values of \texttt{assumeRole} and \texttt{IamInstanceProfileName} in this
sample with the service role ARN and instance profile role you created when \textbf{Getting started
with Automation (p. 281)}.

\begin{verbatim}
{
    "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",
            "inputs":{
                "sourceAMIid":\texttt{your-ami-id},
                "targetAMIName":\texttt{patchedAMI-{{global:DATE_TIME}}}
            }
        }
    ]
}
\end{verbatim}
"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-InstallWindowsUpdates",
  "InstanceIds":[
    "{{ startInstances.InstanceIds }}"
  ],
  "Parameters":{
    "SeverityLevels":"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,
Walkthrough: Patch an AMI and update an Auto Scaling group

The following example builds on the Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store (p. 580) example by adding a step that updates an Auto Scaling group with the newly-patched AMI. This approach ensures that new images are automatically made available to different computing environments that use Auto Scaling groups.

The final step of the Automation workflow in this example uses an AWS Lambda function to copy an existing launch configuration and set the AMI ID to the newly-patched AMI. The Auto Scaling group is then updated with the new launch configuration. In this type of Auto Scaling scenario, users could terminate existing instances in the Auto Scaling group to force a new instance to launch that uses the new image. Or, users could wait and allow scale-in or scale-out events to naturally launch newer instances.

Before You Begin

Complete the following tasks before you begin this example.

- Configure IAM roles for Automation. Systems Manager requires an instance profile role and a service role ARN to process Automation workflows. For more information, see Getting started with Automation (p. 281).
- If you are not familiar with Lambda, we recommend that you create a simple Lambda function by using the Create a Simple Lambda Function topic in the AWS Lambda Developer Guide. The topic will help you understand, in detail, some of the steps required to create a Lambda function.

Task 1: Create an IAM role for AWS Lambda

Use the following procedure to create an IAM service role for AWS Lambda. This role includes the AWSLambdaExecute and AutoScalingFullAccess managed policies. These policies give Lambda permission to create a new Auto Scaling group with the latest, patched AMI using a Lambda function.
To create an IAM service role for Lambda

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
3. On the Select type of trusted entity page, under AWS Service, choose Lambda.
4. In the Select your use case section, choose Lambda, and then choose Next: Permissions.
5. On the Attach permissions policy page, search for AWSLambdaExecute, and then choose the option next to it. Search for AutoScalingFullAccess, and then choose the option next to it.
6. Choose Next: Review.
7. On the Review page, verify that AWSLambdaExecute and AutoScalingFullAccess are listed under Policies.
8. Type a name in the Role name box, and then type a description.
9. Choose Create role. The system returns you to the Roles page.

Task 2: Create an AWS Lambda function

Use the following procedure to create a Lambda function that automatically updates an existing Auto Scaling group with the latest, patched AMI.

To create a Lambda function

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. Choose Create function.
3. Verify that Author from scratch is selected.
4. In the Name field type Automation-UpdateAsg.
5. In the Runtime list, choose Python 2.7.
6. In the Role list, verify that Choose an existing role is selected.
7. In the Existing role list, choose the role you created earlier.
8. Choose Create function. The systems displays a code and configuration page for Automation-UpdateAsg.
9. Make no changes in the Designer section.
10. In the Function code section, delete the pre-populated code in the lambda_function field, and then paste the following code sample.

```python
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. Specify the remaining configuration options on this page.
12. Choose Save.
13. Choose Test.
14. In the Configure test event page, verify that Create new test event is selected.
15. In the Event template list, verify that Hello World is selected.
16. In the Event name field, type a name.
17. Replace the existing sample with the following JSON. Enter an AMI ID and Auto Scaling group.

```json
{
    "newAmiID":"valid AMI ID",
    "targetASG":"name of your Auto Scaling group"
}
```
18. Choose Save.
19. Choose Test. The output states that the Auto Scaling group was successfully updated with a new
launch configuration.
Task 3: Create an Automation document, patch the AMI, and update the Auto Scaling group

Use the following procedure to create and run an Automation document that patches the AMI you specified for the `latestAmi` parameter. The Automation workflow then updates the Auto Scaling group to use the latest, patched AMI.

**To create and run the Automation document**

2. In the navigation pane, choose **Documents**.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (☰) to open the navigation pane, and then choose **Documents** in the navigation pane.
3. Choose **Create automation**.
4. In the **Name** field, type `PatchAmiandUpdateAsg`.
5. Choose the **Editor** tab, and choose the **Edit** button.
6. Choose **OK** when prompted, and delete the placeholder content in the **Document editor** field.
7. In the **Document editor** field, paste the following JSON sample document content.

   **Note**
   You must change the values of `assumeRole` and `IamInstanceProfileName` in this sample with the service role ARN and instance profile role you created when Getting started with Automation (p. 281).

```json
{
  "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"
    },
    "subnetId":{
      "type":"String",
      "description":"The SubnetId where the instance is launched from the sourceAMIid."
    },
    "targetAMIname":{
      "type":"String",
      "description":"Name of new AMI",
      "default":"patchedAMI-{{global:DATE_TIME}}"
    },
    "targetASG":{
      "type":"String",
      "description":"Auto Scaling 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",
"SubnetId": "{{ subnetId }}"
}]
}

{
"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 }}: \"{{ newAmiID }}\"",
"{{ createImage.ImageId }}\"
}
8. Choose **Create document** to save the document.
9. Choose **Automations**, and then choose **Execute automation**.
10. In the **Automation document** list, choose **PatchAmiandUpdateAsg**.
11. In the **Document details** section verify that **Document version** is set to 1.
12. In the **Execution mode** section, choose **Execute the entire automation at once**.
13. Leave the **Targets and Rate Control** option disabled.
14. Specify a Windows AMI ID for **sourceAMIid**, your Auto Scaling group name for **targetASG**, and a value for the **subnetId** input parameter.
15. Choose **Execute automation**.
16. After execution completes, in the Amazon EC2 console, choose **Auto Scaling**, and then choose **Launch Configurations**. Verify that you see the new launch configuration, and that it uses the new AMI ID.
17. Choose **Auto Scaling**, and then choose **Auto Scaling Groups**. Verify that the Auto Scaling group uses the new launch configuration.
18. Terminate one or more instances in your Auto Scaling group. Replacement instances will be launched with the new AMI ID.

**Note**
You can further automate deployment of the new AMI by editing the Lambda function to gracefully terminate instances. You can also invoke your own Lambda function and utilize the ability of AWS CloudFormation to update Auto Scaling groups. For more information, see UpdatePolicy Attribute.

### Using AWS support self-service Automations

This section describes how to run Automations created by the AWS Support team to help you troubleshoot common issues with your AWS resources.

#### Topics
- Walkthrough: Run the EC2Rescue tool on unreachable instances (p. 591)
- Walkthrough: Reset passwords and SSH keys on EC2 instances (p. 596)

### Walkthrough: Run the EC2Rescue tool on unreachable instances

EC2Rescue can help you diagnose and troubleshoot problems on EC2 instances for Linux and Windows Server. You can run the tool manually, as described in Using EC2Rescue for Linux Server and Using EC2Rescue for Windows Server. Or, you can run the tool automatically by using Systems Manager Automation and the **AWSSupport-ExecuteEC2Rescue** document. The **AWSSupport-ExecuteEC2Rescue** document is designed to perform a combination of Systems Manager actions, AWS CloudFormation actions, and Lambda functions that automate the steps normally required to use EC2Rescue.

You can use the **AWSSupport-ExecuteEC2Rescue** document to troubleshoot and potentially remediate different types of operating system (OS) issues. See the following topics for a complete list:

**Windows:** See **Rescue Action** in Using EC2Rescue for Windows Server with the Command Line.
Linux: Some EC2Rescue for Linux modules detect and attempt to remediate issues. For more information, see the aws-ec2rescue-linux documentation for each module on GitHub.

How it works

Troubleshooting an instance with Automation and the AWSSupport-ExecuteEC2Rescue document works as follows:

• You specify the ID of the unreachable instance and run the Automation workflow.
• The system creates a temporary VPC, and then runs a series of Lambda functions to configure the VPC.
• The system identifies a subnet for your temporary VPC in the same Availability Zone as your original instance.
• The system launches a temporary, SSM-enabled helper instance.
• The system stops your original instance, and creates a backup. It then attaches the original root volume to the helper instance.
• The system uses Run Command to run EC2Rescue on the helper instance. EC2Rescue identifies and attempts to fix issues on the attached, original root volume. When finished, EC2Rescue reattaches the root volume back to the original instance.
• The system restarts your original instance, and terminates the temporary instance. The system also terminates the temporary VPC and the Lambda functions created at the start of the automation.

Before you begin

Before you run the following Automation, do the following:

• Copy the instance ID of the unreachable instance. You will specify this ID in the procedure.
• Optionally, collect the ID of a subnet in the same availability zone as your unreachable instance. The EC2Rescue instance will be created in this subnet. If you don’t specify a subnet, then Automation creates a new temporary VPC in your AWS account. Verify that your AWS account has at least one VPC available. By default, you can create five VPCs in a Region. If you already created five VPCs in the Region, the automation fails without making changes to your instance. For more information, see VPC and Subnets.
• Optionally, you can create and specify an AWS Identity and Access Management (IAM) role for Automation. If you don’t specify this role, then Automation runs in the context of the user who ran the automation. For more information about creating roles for Automation, see Running an Automation workflow by using an IAM service role (p. 337).

Granting AWSSupport-EC2Rescue permissions to perform actions on your instances

EC2Rescue needs permission to perform a series of actions on your instances during the Automation execution. These actions invoke the AWS Lambda, IAM, and Amazon EC2 services to safely and securely attempt to remediate issues with your instances. If you have Administrator-level permissions in your AWS account and/or VPC, you might be able to run the automation without configuring permissions, as described in this section. If you don’t have Administrator-level permissions, then you or an administrator must configure permissions by using one of the following options.

• Granting permissions by using IAM policies (p. 592)
• Granting permissions by using an AWS CloudFormation template (p. 594)

Granting permissions by using IAM policies

You can either attach the following IAM policy to your IAM user account, group, or role as an inline policy; or, you can create a new IAM managed policy and attach it to your user account, group, or role. For more information about adding an inline policy to your user account, group, or role see Working With
Inline Policies. For more information about creating a new managed policy, see Working With Managed Policies.

**Note**
If you create a new IAM managed policy, you must also attach the AmazonSSMAutomationRole managed policy to it so that your instances can communicate with the Systems Manager API.

**IAM Policy for AWSSupport-EC2Rescue**

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction",
                "lambda:DeleteFunction",
                "lambda:GetFunction"
            ],
            "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::aws-account-id:role/AWSSupport-EC2Rescue-*",
                "arn:aws:iam::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",
```
Granting permissions by using an AWS CloudFormation template

AWS CloudFormation automates the process of creating IAM roles and policies by using a preconfigured template. Use the following procedure to create the required IAM roles and policies for the EC2Rescue Automation by using AWS CloudFormation.

To create the required IAM roles and policies for EC2Rescue

1. Download AWSSupport-EC2RescueRole.zip and extract the AWSSupport-EC2RescueRole.json file to a directory on your local machine.

2. If your AWS account is in a special partition, edit the template to change the ARN values to those for your partition.

   For example, for the China Regions, change all cases of arn:aws to arn:aws-cn.


4. Choose Create stack, With new resources (standard).

5. On the Create stack page, for Prerequisite - Prepare template, choose Template is ready.

6. For Specify template, choose Upload a template file.

7. Choose file, and then browse to and select the AWSSupport-EC2RescueRole.json file from the directory where you extracted it.

8. Choose Next.

9. On the Specify stack details page, for Stack name field, enter a name to identify this stack, and then choose Next.

10. (Optional) In the Tags area, apply one or more tag key name/value pairs to the stack.

    Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a stack to identify the type of tasks it runs, the types of targets or other resources involved, and the environment it runs in.

11. Choose Next.

12. On the Review page, review the stack details, and then scroll down and choose the I acknowledge that AWS CloudFormation might create IAM resources option.

13. Choose Create stack.

AWS CloudFormation shows the CREATE_IN_PROGRESS status for a few minutes. The status changes to CREATE_COMPLETE after the stack has been created. You can also choose the refresh icon to check the status of the create process.
14. In the **Stacks** list, choose the option button the stack you just created, and then choose the **Outputs** tab.

15. Note the **Value**. The is the ARN of the AssumeRole. You specify this ARN when you run the Automation in the next procedure, *Running the Automation* (p. 595).

**Running the Automation**

**Important**
The following Automation workflow stops the unreachable instance. Stopping the instance can result in lost data on attached instance store volumes (if present). Stopping the instance can also cause the public IP to change, if no Elastic IP is associated.

**To run the AWSSupport-ExecuteEC2Rescue Automation**


2. In the navigation pane, choose **Automation**.
   - or -
   - If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Automation**.

3. Choose **Execute automation**.

4. In the **Automation document** section, choose **Owned by Amazon** from the list.

5. In the documents list, choose the button in the card for *AWSSupport-ExecuteEC2Rescue*, and then choose **Next**.

6. In the **Execute automation document** page, choose **Simple execution**.

7. In the **Document details** section, verify that **Document version** is set to the highest default version. For example, `$DEFAULT` or `3 (default)`.

8. In the **Input parameters** section, specify the following parameters:
   a. For **UnreachableInstanceId**, specify the ID of the unreachable instance.
   b. (Optional) For **EC2RescueInstanceType**, specify an instance type for the EC2Rescue instance. The default instance type is `t2.small`.
   c. For **AutomationAssumeRole**, if you created roles for this Automation by using the AWS CloudFormation procedure described earlier in this topic, then choose the ARN of the AssumeRole that you created in the AWS CloudFormation console.
   d. (Optional) For **LogDestination**, specify an S3 bucket if you want to collect operating system-level logs while troubleshooting your instance. Logs are automatically uploaded to the specified bucket.
   e. For **SubnetId**, specify a subnet in an existing VPC in the same availability zone as the unreachable instance. By default, Systems Manager creates a new VPC, but you can specify a subnet in an existing VPC if you want.

   **Note**
   If you don't see the option to specify a bucket or a subnet ID, verify that you are using the latest **Default** version of the document.

9. (Optional) In the **Tags** area, apply one or more tag key name/value pairs to help identify the execution, for example `Key=Purpose, Value=EC2Rescue`.

10. Choose **Execute**.

The Automation creates a backup AMI as part of the workflow. All other resources created by the Automation workflow are automatically deleted, but this AMI remains in your account. The AMI is named using the following convention:
Backup AMI: AWSSupport-EC2Rescue: UnreachableInstanceId

You can locate this AMI in the Amazon EC2 console by searching on the Automation execution ID.

Walkthrough: Reset passwords and SSH keys on EC2 instances

You can use the AWSSupport-ResetAccess document to automatically reenable local Administrator password generation on EC2 instances for Windows Server, and to generate a new SSH key on EC2 instances for Linux. The AWSSupport-ResetAccess document is designed to perform a combination of Systems Manager actions, AWS CloudFormation actions, and Lambda functions that automate the steps normally required to reset the local administrator password.

You can use Automation with the AWSSupport-ResetAccess document to solve the following problems:

Windows

You lost the EC2 key pair: To resolve this problem, you can use the AWSSupport-ResetAccess document to create a password-enabled AMI from your current instance, launch a new instance from the AMI, and select a key pair you own.

You lost the local Administrator password: To resolve this problem, you can use the AWSSupport-ResetAccess document to generate a new password that you can decrypt with the current EC2 key pair.

Linux

You lost your EC2 key pair, or you configured SSH access to the instance with a key you lost: To resolve this problem, you can use the AWSSupport-ResetAccess document to create a new SSH key for your current instance, which enables you to connect to the instance again.

Note

If your v for Windows Server is configured for Systems Manager, you can also reset your local Administrator password by using EC2Rescue and Run Command. For more information, see Using EC2Rescue for Windows Server with Systems Manager Run Command in the Amazon EC2 User Guide for Windows Instances.

How it works

Troubleshooting an instance with Automation and the AWSSupport-ResetAccess document works as follows:

• You specify the ID of the instance and run the Automation workflow.
• The system creates a temporary VPC, and then runs a series of Lambda functions to configure the VPC.
• The system identifies a subnet for your temporary VPC in the same Availability Zone as your original instance.
• The system launches a temporary, SSM-enabled helper instance.
• The system stops your original instance, and creates a backup. It then attaches the original root volume to the helper instance.
• The system uses Run Command to run EC2Rescue on the helper instance. On Windows, EC2Rescue enables password generation for the local Administrator by using EC2Config or EC2Launch on the attached, original root volume. On Linux, EC2Rescue generates and injects a new SSH key and saves the private key, encrypted, in Parameter Store. When finished, EC2Rescue reattaches the root volume back to the original instance.
• The system creates a new Amazon Machine Image (AMI) of your instance, now that password generation is enabled. You can use this AMI to create a new EC2 instance, and associate a new key pair if needed.
• The system restarts your original instance, and terminates the temporary instance. The system also terminates the temporary VPC and the Lambda functions created at the start of the automation.

• **Windows**: Your instance generates a new password you can decode from the Amazon EC2 console using the current key pair assigned to the instance.

  **Linux**: You can SSH to the instance by using the SSH key stored in Systems Manager Parameter Store as `/ec2rl/openssh/instance_id/key`.

**Before you begin**

Before you run the following Automation, do the following:

• Copy the instance ID of the instance on which you want to reset the Administrator password. You will specify this ID in the procedure.

• Optionally, collect the ID of a subnet in the same availability zone as your unreachable instance. The EC2Rescue instance will be created in this subnet. If you don’t specify a subnet, then Automation creates a new temporary VPC in your AWS account. Verify that your AWS account has at least one VPC available. By default, you can create five VPCs in a Region. If you already created five VPCs in the Region, the automation fails without making changes to your instance. For more information, see VPC and Subnets.

• Optionally, you can create and specify an AWS Identity and Access Management (IAM) role for Automation. If you don’t specify this role, then Automation runs in the context of the user who ran the automation. For more information about creating roles for Automation, see Running an Automation workflow by using an IAM service role (p. 337).

**Granting AWSSupport-EC2Rescue permissions to perform actions on your instances**

EC2Rescue needs permission to perform a series of actions on your instances during the Automation execution. These actions invoke the AWS Lambda, IAM, and Amazon EC2 services to safely and securely attempt to remediate issues with your instances. If you have Administrator-level permissions in your AWS account and/or VPC, you might be able to run the automation without configuring permissions, as described in this section. If you don’t have Administrator-level permissions, then you or an administrator must configure permissions by using one of the following options.

• **Granting permissions by using IAM policies** (p. 597)

• **Granting permissions by using an AWS CloudFormation template** (p. 599)

**Granting permissions by using IAM policies**

You can either attach the following IAM policy to your IAM user account, group, or role as an inline policy; or, you can create a new IAM managed policy and attach it to your user account, group, or role. For more information about adding an inline policy to your user account, group, or role see Working With Inline Policies. For more information about creating a new managed policy, see Working With Managed Policies.

**Note**

If you create a new IAM managed policy, you must also attach the `AmazonSSMAutomationRole` managed policy to it so that your instances can communicate with the Systems Manager API.

**IAM Policy for AWSSupport-ResetAccess**

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [  
```
"lambda:InvokeFunction",
"lambda:DeleteFunction",
"lambda:GetFunction"
],
"Resource": "arn:aws:lambda:*: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::aws-account-id:role/AWSSupport-EC2Rescue-**",
  "arn:aws:iam::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"
}
Granting permissions by using an AWS CloudFormation template

AWS CloudFormation automates the process of creating IAM roles and policies by using a preconfigured template. Use the following procedure to create the required IAM roles and policies for the EC2Rescue Automation by using AWS CloudFormation.

To create the required IAM roles and policies for EC2Rescue

1. Download AWSSupport-EC2RescueRole.zip and extract the AWSSupport-EC2RescueRole.json file to a directory on your local machine.
2. If your AWS account is in a special partition, edit the template to change the ARN values to those for your partition.
   
   For example, for the China Regions, change all cases of arn:aws to arn:aws-cn.
4. Choose Create stack, With new resources (standard).
5. On the Create stack page, for Prerequisite - Prepare template, choose Template is ready.
6. For Specify template, choose Upload a template file.
7. Choose Choose file, and then browse to and select the AWSSupport-EC2RescueRole.json file from the directory where you extracted it.
8. Choose Next.
9. On the Specify stack details page, for Stack name field, enter a name to identify this stack, and then choose Next.
10. (Optional) In the Tags area, apply one or more tag key name/value pairs to the stack.
    
    Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a stack to identify the type of tasks it runs, the types of targets or other resources involved, and the environment it runs in.
11. Choose Next
12. On the Review page, review the stack details, and then scroll down and choose the I acknowledge that AWS CloudFormation might create IAM resources option.
13. AWS CloudFormation shows the CREATE_IN_PROGRESS status for a few minutes. The status changes to CREATE_COMPLETE after the stack has been created. You can also choose the refresh icon to check the status of the create process.
14. In the stack list, choose the option next to the stack you just created, and then choose the Outputs tab.
15. Copy the Value. The is the ARN of the AssumeRole. You will specify this ARN when you run the Automation.

Running the Automation

The following procedure describes how to run the AWSSupport-ResetAccess document by using the AWS Systems Manager console.

Important

The following Automation execution stops the instance. Stopping the instance can result in lost data on attached instance store volumes (if present). Stopping the instance can also cause the public IP to change, if no Elastic IP is associated. To avoid these configuration changes, use Run
Command to reset access. For more information, see Using EC2Rescue for Windows Server with Systems Manager Run Command in the Amazon EC2 User Guide for Windows Instances.

To run the AWSSupport-ResetAccess Automation

2. In the navigation pane, choose Automation.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Automation.

3. Choose Execute automation.
4. In the Automation document section, choose Owned by Amazon from the list.
5. In the documents list, choose the button in the card for AWSSupport-ResetAccess, and then choose Next.
6. In the Execute automation document page, choose Simple execution.
7. In the Document details section, verify that Document version is set to the highest default version. For example, $DEFAULT or 3 (default).
8. In the Input parameters section, specify the following parameters:
   a. For InstanceID, specify the ID of the unreachable instance.
   b. For SubnetId, specify a subnet in an existing VPC in the same availability zone as the instance you specified. By default, Systems Manager creates a new VPC, but you can specify a subnet in an existing VPC if you want.

   Note
   If you don’t see the option to specify a subnet ID, verify that you are using the latest Default version of the document.

   c. For EC2RescueInstanceType, specify an instance type for the EC2Rescue instance. The default instance type is t2.small.
   d. For AssumeRole, if you created roles for this Automation by using the AWS CloudFormation procedure described earlier in this topic, then specify the AssumeRole ARN that you noted in the AWS CloudFormation console.
9. (Optional) In the Tags area, apply one or more tag key name/value pairs to help identify the execution, for example Key=Purpose,Value=ResetAccess.
10. Choose Execute.
11. To monitor the execution progress, choose the running Automation workflow, and then choose the Steps tab. When the execution is finished, choose the Descriptions tab, and then choose View output to view the results. To view the output of individual steps, choose the Steps tab, and then choose View Outputs next to a step.

The Automation creates a backup AMI and a password-enabled AMI as part of the workflow. All other resources created by the Automation workflow are automatically deleted, but these AMIs remain in your account. The AMIs are named using the following conventions:

- Backup AMI: AWSSupport-EC2Rescue:InstanceID
- Password-enabled AMI: AWSSupport-EC2Rescue: Password-enabled AMI from InstanceID

You can locate these AMIs by searching on the Automation execution ID.

For Linux, the new SSH private key for your instance is saved, encrypted, in Parameter Store. The parameter name is /ec2rl/openssh/instance_id/key.
Walkthrough: Using input transformers with Automation

This Systems Manager Automation walkthrough shows how to use the input transformer feature of Amazon CloudWatch Events to extract the instance-id of an EC2 instance from an instance state change event. We use the input transformer to pass that data to the AWS-CreateImage Systems Manager Automation document target as the InstanceId input parameter. The rule is triggered when any instance changes to the stopped state.

For more information about working with input transformers, see Tutorial: Use Input Transformer to Customize What is Passed to the Event Target in the Amazon CloudWatch Events User Guide.

Before You Begin

Verify that you added the required permissions and trust policy for CloudWatch Events to your Systems Manager Automation service role. For more information, see Overview of Managing Access Permissions to Your CloudWatch Events Resources.

To use input transformers with Automation

2. In the navigation pane, choose Events, and then choose Create rule.
3. For Event source, do the following:
   a. Choose Event Pattern.
   b. For Build event pattern to match, choose Events by Service.
   c. For Service Name, choose EC2.
   d. For Event Type, choose EC2 Instance State-change Notification.
   e. Choose Specific state(s) and stopped from the dropdown.
   f. Choose Any instance.
4. For Targets, choose Add target, SSM Automation.
5. For Document, choose AWS-CreateImage.
6. Choose Configure automation parameter(s), Input Transformer.
7. In the first box under Input Transformer, enter "{instance":"$.detail.instance-id"}.
8. In the second box, enter "{InstanceId}:[<instance>]}.
9. Choose Use existing role and choose your Automation service role.
10. Choose Configure details.
11. Enter a Name and Description for the rule, and choose Create rule.

Walkthrough: Using Automation with Jenkins

If your organization uses Jenkins software in a CI/CD pipeline, you can add Automation as a post-build step to pre-install application releases into Amazon Machines Images (AMIs). You can also use the Jenkins scheduling feature to call Automation and create your own operating system (OS) patching cadence.

The example below shows how to invoke Automation from a Jenkins server that is running either on-premises or in Amazon EC2. For authentication, the Jenkins server uses AWS credentials based on an AWS Identity and Access Management (IAM) user that you create in the example. If your Jenkins server is running in Amazon EC2, you can also authenticate it using an IAM instance profile role.

Note
Be sure to follow Jenkins security best-practices when configuring your instance.
Before You Begin

Complete the following tasks before you configure Automation with Jenkins.

- Complete the Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store (p. 580) example. The following example uses the UpdateMyLatestWindowsAmi automation document created in that example.

- Configure IAM roles for Automation. Systems Manager requires an instance profile role and a service role ARN to process Automation workflows. For more information, see Getting started with Automation (p. 281).

- After you configure IAM roles for Automation, use the following procedure to create an IAM user account for your Jenkins server. The Automation workflow uses the IAM user account's Access key and Secret key to authenticate the Jenkins server during execution.

To create a user account for the Jenkins server

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy.
3. Choose the JSON tab.
4. Replace the default content with the following. Be sure to replace us-west-2 and 123456789012 with the Region and account you want to use.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": [
            ]
        }
    ]
}
```

5. Choose Review policy.
6. On the Review policy page, for Name, enter a name for the inline policy, such as JenkinsPolicy.
7. Choose Create policy.
8. In the navigation pane, choose Users.
9. Choose Add user.
10. In the Set user details section, specify a user name (for example, Jenkins).
11. In the Select AWS access type section, choose Programmatic Access.
12. Choose Next:Permissions.
13. In the Set permissions for section, choose Attach existing policies directly.
14. In the filter field, type the name of the policy you created earlier.
15. Select the check box next to the policy, and then choose Next: Tags.
16. (Optional) Add one or more tag key-value pairs to organize, track, or control access for this user, and then choose Next: Review.
17. Verify the details, and then choose Create.
18. Copy the access and secret keys to a text file. You will specify these credentials in the next procedure.
Use the following procedure to configure the AWS CLI on your Jenkins server.

**To configure the Jenkins server for Automation**

1. If it's not already installed, download the AWS CLI to your Jenkins server. For more information, see Installing the AWS Command Line Interface.
2. In a terminal window on your Jenkins server, run the following commands to configure the AWS CLI.
   
   ```
   sudo su - jenkins
   aws configure
   ```

   For information, see Install or upgrade AWS command line tools (p. 57).
3. When prompted, enter the AWS Access key and Secret key you received when you created the Jenkins user in IAM. Specify a default region. For more information about configuring the AWS CLI see Configuring the AWS Command Line Interface.

Use the following procedure to configure your Jenkins project to invoke Automation.

**To configure your Jenkins server to invoke Automation**

1. Open the Jenkins console in a web browser.
2. Choose the project that you want to configure with Automation, and then choose **Configure**.
3. On the **Build** tab, choose **Add Build Step**.
4. Choose **Execute shell** or **Execute Windows batch command** (depending on your operating system).
5. In the **Command** box, run an AWS CLI command like the following:

   ```
   aws --region the AWS Region of your source AMI ssm start-automation-execution --
   document-name your document name --parameters parameters for the document
   ```

The following example command uses the **UpdateMyLatestWindowsAmi** document and the Systems Manager Parameter **latestAmi** created in Walkthrough: Simplify AMI patching using Automation, AWS Lambda, and Parameter Store (p. 580):

```
aws --region region-id ssm start-automation-execution \
--document-name UpdateMyLatestWindowsAmi \
--parameters "sourceAMIId='{{ssm:latestAmi}}'"
```

In Jenkins, the command looks like the example in the following screenshot.
6. In the Jenkins project, choose Build Now. Jenkins returns output similar to the following example.

```
Console Output

Started by user admin
Building in workspace /var/lib/jenkins/workspace/build-AWI
[Build #1] 4.351s to run /opt/local/bin/jenkinsw/1259512977444166019.sh
+ aws --region us-east-1 ssm start-automation-execution --document-name UpdateMyLatestWindows --parameters '{"sourceNId":"$(ssm:getParameter|Name=LatestWindows|WithDecryption=true|WithVersion=true)"}'

"AutomationExecutionId": "5beef73a-ef06-1f56-9502-9d6daa849f3"

Finished: SUCCESS
```

Walkthrough: Using Document Builder to create a custom Automation document

The following walkthrough shows how to use Document Builder in the Systems Manager Automation console to create a custom Automation document and then run the custom Automation document.

The first step of the Automation document you create runs a script to launch an Amazon Elastic Compute Cloud (EC2) instance. The second step runs another script to monitor for the instance status check to change to ok. Then, an overall status of Success is reported for the automation execution.

Before You Begin

Before you begin this walkthrough, do the following:

- Verify that you have administrator privileges, or that you have been granted the appropriate permissions to access Systems Manager in AWS Identity and Access Management (IAM).
  
  For information, see Verifying user access for Automation workflows (p. 281).

- Verify that you have an IAM service role for Automation (also known as an assume role) in your AWS account. The role is required because this walkthrough uses the `aws:executeScript` action.
  
  For information about creating this role, see Configuring a service role (assume role) access for Automation workflows (p. 281).

- Verify that you have permission to launch EC2 instances.

  For information about the IAM service role requirement for running `aws:executeScript`, see Permissions for running Automation executions (p. 358).

  For information, see IAM and Amazon EC2 in the Amazon EC2 User Guide for Linux Instances.

Topics

- Step 1: Create the custom Automation document (p. 604)
- Step 2: Run the custom Automation document (p. 609)

Step 1: Create the custom Automation document

Use the following procedure to create a custom Automation document that launches an EC2 instance and waits for the instance status check to change to ok.

**Tip**

If you copy and paste values from this walkthrough into Document Builder, such as parameter names and handler names, make sure to delete any leading or trailing spaces added to the text value you enter.
To create a custom Automation document using Document Builder

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon ( ) to open the navigation pane, and then choose Documents in the navigation pane.
3. Choose Create automation.
4. For Name, type this descriptive name for the document: LaunchInstanceAndCheckStatus.
5. (Optional) For Document description, replace the default text with a description for this document, using Markdown. The following is an example.

```markdown
#Title: LaunchInstanceAndCheckStatus
------
**Purpose**: This Automation document first launches an EC2 instance using the AMI ID provided in the parameter `imageId`. The second step of this document continuously checks the instance status check value for the launched instance until the status `ok` is returned.

##Parameters:
------
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>assumeRole</td>
<td>String</td>
<td>(Optional) The ARN of the role that allows Automation to perform actions on your behalf.</td>
<td>-</td>
</tr>
<tr>
<td>imageId</td>
<td>String</td>
<td>(Optional) The AMI ID to use for launching the instance. The default value uses the latest Amazon Linux AMI ID available.</td>
<td>{{ ssm:/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2 }}</td>
</tr>
</tbody>
</table>
```

6. For Assume role, enter the ARN of the IAM service role for Automation (Assume role) for the automation execution, in the format `arn:aws:iam::111122223333:role/AutomationServiceRole`. Substitute your AWS account ID for 111122223333.

   The role you specify is used to provide the permissions needed to start the automation execution.

   **Important**
   For Automation documents not owned by Amazon that use the `aws:executeScript` action, a role must be specified. For information, see Permissions for running Automation executions (p. 358).

7. Expand Input parameters and do the following.
   1. For Parameter name, enter imageId.
   2. For Type, choose String.
   3. For Required, choose No.
   4. For Default value, enter the following.

   ```
   {{ ssm:/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2 }}
   ```

   **Note**
   This value launches an EC2 instance using the latest Amazon Linux Amazon Machine Image (AMI) ID. If you want to use a different AMI, replace the value with your AMI ID.

   5. For Description, enter the following.

   (Optional) The AMI ID to use for launching the instance. The default value uses the latest released Amazon Linux AMI ID.
8. Choose **Add a parameter** to create the second parameter, **tagValue**, and enter the following.

1. For **Parameter name**, enter **tagValue**.
2. For **Type**, choose **String**.
3. For **Required**, choose **No**.
4. For **Default value**, enter **LaunchedBySsmAutomation**. This adds the tag key-pair value Name:LaunchedBySsmAutomation to the instance.
5. For **Description**, enter the following.

   (Optional) The tag value to add to the instance. The default value is LaunchedBySsmAutomation.

9. Choose **Add a parameter** to create the third parameter, **instanceType**, and enter the following information.

1. For **Parameter name**, enter **instanceType**.
2. For **Type**, choose **String**.
3. For **Required**, choose **No**.
4. For **Default value**, enter **t2.micro**.
5. For **Parameter description**, enter the following.

   (Optional) The instance type to use for the instance. The default value is t2.micro.

10. Expand **Target type** and choose "/".
11. (Optional) Expand **Document tags** to apply resource tags to your Automation document. For **Tag key**, enter **Purpose**, and for **Tag value**, enter **LaunchInstanceAndCheckState**.
12. In the **Step 1** section, complete the following steps.

1. For **Step name**, enter this descriptive step name for the first step of the automation workflow: **LaunchEc2Instance**.
2. For **Action type**, choose **Run a script** (**aws:executeScript**).
3. For **Description**, enter a description for the automation step, such as the following.

   **About This Step**
   
   This step first launches an EC2 instance using the `aws:executeScript` action and the provided script.

4. Expand **Inputs**.
5. For **Runtime**, choose the runtime language to use to run the provided script.
6. For **Handler**, enter **launch_instance**. This is the function name declared in the following script.

   **Note**
   
   This is not required for PowerShell.

7. For **Script**, replace the default contents with the following. Be sure to match the script with the corresponding runtime value.

   **Python**

   ```python
def launch_instance(events, context):
    import boto3
    ec2 = boto3.client('ec2')
    
    image_id = events['image_id']
    tag_value = events['tag_value']
    instance_type = events['instance_type']
```
```
tag_config = {'ResourceType': 'instance', 'Tags': [{ 'Key': 'Name', 'Value': tag_value}]}res = ec2.run_instances(ImageId=image_id, InstanceType=instance_type, MaxCount=1, MinCount=1, TagSpecifications=[tag_config])instance_id = res['Instances'][0]['InstanceId']print('[INFO] 1 EC2 instance is successfully launched', instance_id)return { 'InstanceId' : instance_id }
```

PowerShell

```
Install-Module AWS.Tools.EC2 -Force
Import-Module AWS.Tools.EC2
$payload = $env:InputPayload | ConvertFrom-Json
$imageid = $payload.image_id
$tagvalue = $payload.tag_value
$instanceType = $payload.instance_type
$type = New-Object Amazon.EC2.InstanceType -ArgumentList $instanceType
$resource = New-Object Amazon.EC2.ResourceType -ArgumentList 'instance'
$tag = @{Key='Name';Value=$tagValue}
$tagSpecs = New-Object Amazon.EC2.Model.TagSpecification
$tagSpecs.ResourceType = $resource
$tagSpecs.Tags.Add($tag)
$res = New-EC2Instance -ImageId $imageId -MinCount 1 -MaxCount 1 -InstanceType $type -TagSpecification $tagSpecs
return @{'InstanceId'=$res.Instances.InstanceId}
```

8. Expand Additional inputs.
9. For Input name, choose InputPayload. For Input value, enter the following YAML data.

```
image_id: "{{ imageId }}"
tag_value: "{{ tagValue }}"
instance_type: "{{ instanceType }}"
```

13. Expand Outputs and do the following:

- For Name, enter payload.
- For Selector, enter $.Payload.
- For Type, choose StringMap.

14. Choose Add step to add a second step to the Automation document. The second step queries the status of the instance launched in Step 1 and waits until the status returned is ok.

15. In the Step 2 section, do the following.
1. For **Step name**, enter this descriptive name for the second step of the automation workflow: `WaitForInstanceStatusOk`.

2. For **Action type**, choose `Run a script (aws:executeScript)`.

3. For **Description**, enter a description for the automation step, such as the following.

   **About This Step**
   The script continuously polls the instance status check value for the instance launched in Step 1 until the `'ok'` status is returned.

4. For **Runtime**, choose the runtime language to be used for executing the provided script.

5. For **Handler**, enter `poll_instance`. This is the function name declared in the following script.

   **Note**
   This is not required for PowerShell.

6. For **Script**, replace the default contents with the following. Be sure to match the script with the corresponding runtime value.

   **Python**

   ```python
def poll_instance(events, context):
    import boto3
    import time

    ec2 = boto3.client('ec2')
    instance_id = events['InstanceId']

    print('[INFO] Waiting for instance status check to report ok', instance_id)
    instance_status = "null"

    while True:
        res = ec2.describe_instance_status(InstanceIds=[instance_id])

        if len(res['InstanceStatuses']) == 0:
            print("Instance status information is not available yet")
            time.sleep(5)
            continue

        instance_status = res['InstanceStatuses'][0]['InstanceStatus']['Status']
        print('[INFO] Polling to get status of the instance', instance_status)

        if instance_status == 'ok':
            break

        time.sleep(10)

    return {'Status': instance_status, 'InstanceId': instance_id}
```

   **PowerShell**

   ```powershell
   Install-Module AWS.Tools.EC2 -Force
   $inputPayload = $env:InputPayload | ConvertFrom-Json
   $instanceId = $inputPayload.payload.InstanceId
   $status = Get-EC2InstanceStatus -InstanceId $instanceId
   ```
7. Expand Additional inputs.
8. For Input name, choose InputPayload. For Input value, enter the following:

```powershell
{{ LaunchEc2Instance.payload }}
```

16. Choose **Create automation** to save the document.

**Step 2: Run the custom Automation document**

Use the following procedure to run the custom Automation document created in Step 1. The custom Automation document launches an EC2 instance and waits for the instance check to change to the *ok* status.

**To run the custom Automation document**

2. In the navigation pane, choose **Automation**, and then choose **Execute automation**.
3. In the Automation document list, choose the **Owned by me** tab and then choose the button next to the custom Automation document you created, **LaunchInstanceAndCheckStatus**.
4. In the Document details section, for Document version, verify that Default version at runtime is selected.
5. Choose **Next**.
6. At the top of the Execute automation document page, verify that Simple execution is selected.
7. Choose **Execute**.
8. After both steps in the automation workflow complete, in the Executed steps area, choose the step ID of a step to view steps details, including any step output.

**Note**

- It can take several minutes for the *ok* status to be returned.

9. (Optional) Unless you plan to use the EC2 instance created by this walkthrough for other purposes, you can terminate the instance. For information, see **Terminate Your Instance** in the [Amazon EC2 User Guide for Linux Instances](https://docs.aws.amazon.com/AWSEC2/latest/UG/ TerminatingInstances.html).

You can identify the instance by the name **LaunchedBySsmAutomation** that you tagged it with in Step 1: Create the custom Automation document (p. 604).

**Troubleshooting Systems Manager Automation**

Use the following information to help you troubleshoot problems with the Automation service. This topic includes specific tasks to resolve issues based on Automation error messages.

**Topics**
- Common Automation errors (p. 610)
- Automation execution failed to start (p. 610)
• Execution started, but status is failed (p. 611)
• Execution started, but timed out (p. 613)

Common Automation errors

This section includes information about common Automation errors.

VPC not defined 400

By default, when Automation runs either the AWS-UpdateLinuxAmi document or the AWS-UpdateWindowsAmi document, the system creates a temporary instance in the default VPC (172.30.0.0/16). If you deleted the default VPC, you will receive the following error:

VPC not defined 400

To solve this problem, you must specify a value for the SubnetId input parameter.

Automation execution failed to start

An Automation execution can fail with an access denied error or an invalid assume role error if you have not properly configured IAM users, roles, and policies for Automation.

Access denied

The following examples describe situations when an Automation execution failed to start with an access denied error.

Access Denied to Systems Manager API


• Possible cause 1: The IAM user attempting to start the Automation execution does not have permission to invoke the StartAutomationExecution API. To resolve this issue, attach the required IAM policy to the user account that was used to start the execution. For more information, see Task 3: Configure user access to Automation (p. 285).

• Possible cause 2: The IAM user attempting to start the Automation execution has permission to invoke the StartAutomationExecution API, but does not have permission to invoke the API by using the specific Automation document. To resolve this issue, attach the required IAM policy to the user account that was used to start the execution. For more information, see Task 3: Configure user access to Automation (p. 285).

Access Denied Because of Missing PassRole Permissions

Error message: 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-xxxxxxxxxxxx)

The IAM user attempting to start the Automation execution does not have PassRole permission for the assume role. To resolve this issue, attach the iam:PassRole policy to the role of the IAM user attempting to start the Automation execution. For more information, see Task 2: Attach the iam:PassRole policy to your Automation role (p. 285).
Invalid assume role

When you run an Automation, an assume role is either provided in the document or passed as a parameter value for the document. Different types of errors can occur if the assume role is not specified or configured properly.

Malformed Assume Role

Error message: The format of the supplied assume role ARN is invalid. The assume role is improperly formatted. To resolve this issue, verify that a valid assume role is specified in your Automation document or as a runtime parameter when running the Automation.

Assume Role Can’t Be Assumed

Error message: The defined assume role is unable to be assumed.

   (Service: AWSSimpleSystemsManagement; Status Code: 400; Error Code: InvalidAutomationExecutionParametersException; Request ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)

- Possible cause 1: The assume role does not exist. To resolve this issue, create the role. For more information, see the section called “Getting started with Automation” (p. 281). Specific details for creating this role are described in the following topic, Task 1: Create a service role for Automation (p. 283).

- Possible cause 2: The assume role does not have a trust relationship with the Systems Manager service. To resolve this issue, create the trust relationship. For more information, see I Can’t Assume A Role in the IAM User Guide.

Execution started, but status is failed

Action-specific failures

Automation documents contain steps and steps run in order. Each step invokes one or more AWS service APIs. The APIs determine the inputs, behavior, and outputs of the step. There are multiple places where an error can cause a step to fail. Failure messages indicate when and where an error occurred.

To see a failure message in the EC2 console, choose the View Outputs link of the failed step. To see a failure message from the AWS CLI, call get-automation-execution and look for the FailureMessage attribute in a failed StepExecution.

In the following examples, a step associated with the aws:runInstance action failed. Each example explores a different type of error.

Missing Image

Error message: 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-xxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

The aws:runInstances action received input for an ImageId that doesn't exist. To resolve this problem, update the automation document or parameter values with the correct AMI ID.

Assume Role Policy Doesn’t Have Sufficient Permissions

Error message: 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: xxxxxxx (Service: AmazonEC2; Status Code: 403; Error Code: UnauthorizedOperation; Request ID: xxxxxxxxxx-xxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx)]. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

The assume role doesn’t have sufficient permission to invoke the RunInstances API on EC2 instances. To resolve this problem, attach an IAM policy to the assume role that has permission to invoke the RunInstances API. For more information, see the Method 2: Use IAM to configure roles for Automation (p. 283).

Unexpected State

Error message: Step fails when it is verifying launched instance(s) are ready to be used. Instance i-xxxxxxxx entered unexpected state: shutting-down. Please refer to Automation Service Troubleshooting Guide for more diagnosis details.

• Possible cause 1: There is a problem with the instance or the Amazon EC2 service. To resolve this problem, login to the instance or review the instance system log to understand why the instance started shutting down.
• Possible cause 2: The user data script specified for the aws:runInstances action has a problem or incorrect syntax. Verify the syntax of the user data script. Also, verify that the user data scripts doesn’t shut down the instance, or invoke other scripts that shut down the instance.

Action-Specific Failures Reference

When a step fails, the failure message might indicate which service was being invoked when the failure occurred. The following table lists the services invoked by each action. The table also provides links to information about each service.

<table>
<thead>
<tr>
<th>Action</th>
<th>AWS Service(s) invoked by this action</th>
<th>For information about this service</th>
<th>Troubleshooting content</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws:runInstances</td>
<td>Amazon EC2</td>
<td>Amazon EC2 User Guide for Linux Instances</td>
<td>Troubleshooting EC2 Instances</td>
</tr>
<tr>
<td>aws:changeInstanceState</td>
<td>Amazon EC2</td>
<td>Amazon EC2 User Guide for Linux Instances</td>
<td>Troubleshooting EC2 instances</td>
</tr>
<tr>
<td>aws:runCommand</td>
<td>Systems Manager</td>
<td>AWS Systems Manager Run Command (p. 829)</td>
<td>Troubleshooting Systems Manager Run Command (p. 862)</td>
</tr>
<tr>
<td>aws:createImage</td>
<td>Amazon EC2</td>
<td>Amazon Machines Images</td>
<td></td>
</tr>
<tr>
<td>aws:createStack</td>
<td>AWS CloudFormation</td>
<td>AWS CloudFormation User Guide</td>
<td>Troubleshooting AWS CloudFormation</td>
</tr>
<tr>
<td>aws:deleteStack</td>
<td>AWS CloudFormation</td>
<td>AWS CloudFormation User Guide</td>
<td>Troubleshooting AWS CloudFormation</td>
</tr>
<tr>
<td>aws:deleteImage</td>
<td>Amazon EC2</td>
<td>Amazon Machines Images</td>
<td></td>
</tr>
<tr>
<td>aws:copyImage</td>
<td>Amazon EC2</td>
<td>Amazon Machines Images</td>
<td></td>
</tr>
</tbody>
</table>
### AWS Systems Manager Change Calendar

Systems Manager Change Calendar lets you set up date and time ranges when actions you specify (for example, in Systems Manager Automation (p. 278) documents) may or may not be performed in your AWS account. In Change Calendar, these ranges are called **events**. When you create a Change Calendar entry, you are creating a Systems Manager document (p. 1031) of the type `ChangeCalendar`. In Change Calendar, the document stores iCalendar 2.0 data in plaintext format. Events that you add to the Change Calendar entry become part of the document.

A Change Calendar entry can be one of two types:

- **DEFAULT_OPEN**, or Open by default
  
  When a calendar entry is open by default, actions can run by default, but are blocked from running during associated events. During events, the state of a DEFAULT_OPEN calendar is **CLOSED**.

- **DEFAULT_CLOSED**, or Closed by default
  
  When a calendar entry is closed by default, actions that are tracking Change Calendar do not run by default, but can run during events associated with the calendar entry. During events, the state of a DEFAULT_CLOSED calendar is **OPEN**.

### Who should use Change Calendar?

- Any AWS customer who creates or runs Systems Manager Automation documents.
Administrators who are responsible for keeping the configurations of AWS Systems Manager managed instances consistent, stable, and functional.

Benefits of Change Calendar

The following are some benefits of Systems Manager Change Calendar.

- **Review changes before they're applied**

  A Change Calendar entry can help ensure that potentially destructive Automation changes to your environment are reviewed before they're applied.

- **Apply changes only during appropriate times**

  Change Calendar entries help keep your environment stable during event times. For example, you can create a Change Calendar entry to block changes when you expect high demand on your resources, such as during a conference or public marketing promotion. A calendar entry can also block changes when you expect limited administrator support, such as during vacations or holidays. You can use a calendar entry to allow changes except for certain times of the day or week when there is limited administrator support to troubleshoot failed actions or deployments.

- **Get the current or upcoming state of the calendar**

  You can run the Systems Manager `GetCalendarState` API operation to show you the current state of the calendar, the state at a specified time, or the next time that the calendar state is scheduled to change.

Topics

- Getting started with Change Calendar (p. 614)
- Working with Change Calendar (p. 615)
- Add Change Calendar dependencies to Automation documents (p. 619)

Getting started with Change Calendar

Complete the following before using Change Calendar.

Install latest command line tools

Install the latest command line tools to get state information about calendars.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS CLI (Optional)</td>
<td>To use the AWS CLI to get state information about calendars, install the newest release of the AWS CLI on your local computer. For more information about how to install or upgrade the CLI, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide.</td>
</tr>
<tr>
<td>AWS Tools for PowerShell (Optional)</td>
<td>To use the Tools for PowerShell to get state information about calendars, install the</td>
</tr>
</tbody>
</table>
**Set up permissions**

To create, update, or delete a Change Calendar entry, including adding and removing events from the entry, a policy attached to your AWS Identity and Access Management (IAM) user or service role must allow the following actions.

- `ssm:CreateDocument`
- `ssm:DescribeDocument`
- `ssm:UpdateDocument`
- `ssm:DeleteDocument`

To get information about the current or upcoming state of the calendar, a policy attached to your AWS Identity and Access Management (IAM) user or service role must allow the following action.

- `ssm:GetCalendarState`

If your IAM user account, group, or role is assigned administrator permissions, then you have access to Change Calendar. If you don’t have administrator permissions, then an administrator must give you permission by assigning the `AmazonSSMFullAccess` managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

Change Calendar entries that are owned by (that is, created by) accounts other than yours are read-only, even if they are shared with your account.

**Working with Change Calendar**

You can use the AWS Systems Manager console to add, manage, or delete Change Calendar entries. Use the `GetCalendarState` API or `get-calendar-state` AWS CLI command to get information about the state of Change Calendar at a specific time.

**Topics**

- Create a Change Calendar entry (p. 616)
- Create a Change Calendar event (p. 616)
- Update a Change Calendar event (p. 617)
- Delete a Change Calendar event (p. 617)
- Update a Change Calendar entry (p. 618)
- Share a Change Calendar entry (p. 618)
- Delete a Change Calendar entry (p. 618)
- Get the state of the Change Calendar (p. 619)
Create a Change Calendar entry

When you create an AWS Systems Manager Change Calendar entry, you are creating a Systems Manager document that uses the text format.

Create a Change Calendar entry (console)

2. In the navigation pane, choose Change Calendar.
3. On the Change Calendar home page, choose Create calendar.
4. On the Create calendar page, in Calendar details, enter a name for your calendar entry. Calendar entry names can contain letters, numbers, periods, dashes, and underscores. The name should be specific enough to identify the purpose of the calendar entry at a glance. An example is support-off-hours. You cannot update this name after you create the calendar entry.
5. In Description, enter a description for your calendar entry.
6. In Calendar type, choose one of the following.
   - Open by default - the calendar is open (Automation actions can run until an event starts), then closed for the duration of an associated event.
   - Closed by default - the calendar is closed (Automation actions cannot run until an event starts) but open for the duration of an associated event.
7. Choose Create calendar.

After the calendar entry is created, Systems Manager displays your calendar entry in the Change Calendar list. The columns show the calendar version, and the calendar owner's AWS account number. Your calendar entry cannot prevent or allow any actions until you add at least one event. For information about how to add an event, see Create a Change Calendar event (p. 616).

Create a Change Calendar event

When you add an event to a Change Calendar entry, you are specifying a period of time during which the default action of the calendar entry is suspended. For example, if the calendar entry type is closed by default, the calendar is open to changes during events. In this release, you can only create a Change Calendar event by using the console. Events are added to the Systems Manager Change Calendar document that you create when you create a Change Calendar entry.

2. In the navigation pane, choose Change Calendar.
3. In the Change Calendar list, choose the name of the calendar entry to which you want to add an event.
4. On the calendar entry's details page, choose Create event.
5. On the Create scheduled event page, in Event details, enter a display name for your event. Event names can contain letters, numbers, periods, dashes, and underscores. The name should be specific enough to identify the purpose of the event. An example is nighttime-hours. You cannot update this name after you create the event.
6. In Description, enter a description for your event. For example, The support team is not available during these hours.
7. In Event start date, enter or choose a day in the format MM/DD/YYYY to start the event, and enter a time on the specified day in the format hh:mm:ss (hours, minutes, and seconds) to start the event.
8. In Event end date, enter or choose a day in the format MM/DD/YYYY to end the event, and enter a time on the specified day in the format hh:mm:ss (hours, minutes, and seconds) to end the event.
9. In **Schedule time zone**, choose a time zone that applies to the start and end times of the event. You can enter part of a city name or time zone difference from Greenwich Mean Time (GMT) to find a time zone faster. The default is Universal Coordinated Time (UTC).

10. To create an event that recurs daily, weekly, or monthly, turn on **Recurrence**.

11. Choose **Create scheduled event**. The new event is added to your calendar entry, and is displayed on the **Events** tab of the calendar entry's details page.

## Update a Change Calendar event

### Update a Change Calendar event (console)

2. In the navigation pane, choose **Change Calendar**.
3. In the **Change Calendar** list, choose the name of the calendar entry for which you want to edit an event.
4. On the calendar entry's details page, choose **Events**.
5. In the calendar page, choose the event that you want to edit. Use the buttons on the upper left to move back or forward one year, or back or forward one month. Change the time zone, if required, by choosing the correct time zone from the drop-down list.
6. In **Event details**, you can change the **Description** text. You cannot edit the name of the event.
7. To change the **Event start date** value, enter or choose a day in the format **DD/MM/YYYY** to start the event, and enter a time on the specified day in the format **hh:mm:ss** (hours, minutes, and seconds) to start the event.
8. To change the **Event end date** value, enter or choose a day in the format **DD/MM/YYYY** to end the event, and enter a time on the specified day in the format **hh:mm:ss** (hours, minutes, and seconds) to end the event.
9. To change the **Schedule time zone** value, choose a time zone that applies to the start and end times of the event. You can enter part of a city name or time zone difference from Greenwich Mean Time (GMT) to find a time zone faster. The default is Universal Coordinated Time (UTC).
10. To update an event to recur daily, weekly, or monthly, turn on **Recurrence**.
11. Choose **Update scheduled event**. Your changes are displayed on the **Events** tab of the calendar entry's details page. Choose the event that you updated to view your changes.

## Delete a Change Calendar event

You can delete an event by using the AWS Management Console.

### Delete a Change Calendar event (console)

You can delete one event at a time by using the AWS Management Console.

2. In the navigation pane, choose **Change Calendar**.
3. In the **Change Calendar** list, choose the name of the calendar entry from which you want to delete an event.
4. On the calendar entry's details page, choose **Events**.
5. In the calendar page, choose the event that you want to delete. Use the buttons on the upper left to move the calendar back or forward one year, or back or forward one month. Change the time zone, if required, by choosing the correct time zone from the drop-down list.
6. On the **Event details** page, choose **Delete**. When you are prompted to confirm that you want to delete the event, choose **Delete**.
Update a Change Calendar entry

Update a Change Calendar entry (console)

You can update a Change Calendar entry's description, but not its name. Although you can change the default state of a calendar entry, be aware that this reverses the behavior of change actions during events that are associated with the calendar entry. For example, if you change the state of a calendar from **Open by default** to **Closed by default**, unwanted changes might be made during event periods when the users who created the associated events are not expecting changes.

When you update a Change Calendar entry, you are editing the Systems Manager Change Calendar document that you created when you created the Change Calendar entry.

2. In the navigation pane, choose **Change Calendar**.
3. In the **Change Calendar** list, choose the name of the calendar entry that you want to update.
4. On the calendar entry's details page, choose **Edit**.
5. In **Description**, you can change the description text. You cannot edit the name of a Change Calendar entry.
6. To change the calendar state, in **Calendar type**, choose a different value. Be aware that this reverses the behavior of change actions during events that are associated with the calendar entry. Before you change the calendar type, you should verify with other Change Calendar users that changing the calendar type does not allow unwanted changes during events that they have created.
   - **Open by default** - the calendar is open (Automation actions can run until an event starts), then closed for the duration of an associated event.
   - **Closed by default** - the calendar is closed (Automation actions cannot run until an event starts) but open for the duration of an associated event.
7. Choose **Save**.

Your calendar entry cannot prevent or allow any actions until you add at least one event. For information about how to add an event, see **Create a Change Calendar event** (p. 616).

Share a Change Calendar entry

You can share a Change Calendar entry with other AWS accounts by using the AWS Management Console. When you share a calendar, the calendar is read-only to users in the shared account.

2. In the navigation pane, choose **Change Calendar**.
3. In the **Change Calendar** list, choose the name of the calendar entry that you want to share.
4. On the calendar entry's details page, choose **Share**.
5. In **Share calendar**, for **Account ID**, enter the ID number of a valid AWS account, and then choose **Share**.

Users of the shared account can read the Change Calendar, but cannot make changes.

Delete a Change Calendar entry

You can delete a Change Calendar entry by using either the AWS Management Console or the AWS CLI. Deleting a Change Calendar entry deletes all associated events.
Delete a Change Calendar entry (console)

2. In the navigation pane, choose Change Calendar.
3. In the Change Calendar list, choose the name of the calendar entry that you want to delete.
4. On the calendar entry's details page, choose Delete. When you are prompted to confirm that you want to delete the calendar entry, choose Delete.

Get the state of the Change Calendar

You can get the overall state of the calendar, or the state at a specific time. You can also show the next time that the calendar state changes from OPEN to CLOSED, or vice versa.

You can do this task only by using the GetCalendarState API. The procedure in this section uses the AWS CLI.

- Run the following command to show the state of one or more calendar entries at a specific time. The --calendar-names parameter is required, but --at-time is optional.

```
aws ssm get-calendar-state --calendar-names
["Calendar_name_or_document_ARN_1","Calendar_name_or_document_ARN_2"] --at-time "ISO_8601_time_format"
```

The following is an example.

```
```

The results show the state of the calendar (whether the calendar is of type DEFAULT_OPEN or DEFAULT_CLOSED) for the specified calendar entries that are owned by or shared with your account, at the time specified as the value of --at-time, and the time of the next transition. If you do not add the --at-time parameter, the current time is used.

Add Change Calendar dependencies to Automation documents

To make Automation actions adhere to Systems Manager Change Calendar, add a step in an Automation document that uses the aws:assertAwsResourceProperty (p. 395) action. Configure the action to run GetCalendarState to verify that a specified calendar entry is in the state that you want (OPEN or CLOSED). The Automation document is only allowed to continue to the next step if the calendar state is OPEN. The following is a YAML-based sample excerpt of an Automation document that cannot advance to the next step, LaunchInstance, unless the calendar state matches OPEN, the state specified in DesiredValues.

```
mainSteps:
- name: MyCheckCalendarStateStep
  action: 'aws:assertAwsResourceProperty'
  inputs:
    Service: ssm
    Api: GetCalendarState
```
AWS Systems Manager Maintenance Windows

AWS Systems Manager Maintenance Windows let you define a schedule for when to perform potentially disruptive actions on your instances such as patching an operating system, updating drivers, or installing software or patches. Maintenance Windows also lets you schedule actions on numerous other AWS resource types, such as Amazon Simple Storage Service (Amazon S3) buckets, Amazon Simple Queue Service (Amazon SQS) queues, AWS Key Management Service (AWS KMS) keys, and many more. For a full list of supported resource types that you can include in a maintenance window target, see Supported Resources for AWS Resource Groups in the AWS Resource Groups User Guide.

Each maintenance window has a schedule, a maximum duration, a set of registered targets (the instances or other AWS resources that are acted upon), and a set of registered tasks. You can add tags to your maintenance windows when you create or update them. (Tags are keys that help identify and sort your resources within your organization.) You can also specify dates that a maintenance window should not run before or after, and you can specify the international time zone on which to base the maintenance window schedule. (For an explanation of how the various schedule-related options for maintenance windows relate to one another, see Reference: Maintenance window scheduling and active period options (p. 693).)

Maintenance windows support running four types of tasks:

- Systems Manager Run Command commands
  For more information about Run Command, see AWS Systems Manager Run Command (p. 829).
- Systems Manager Automation workflows
  For more information about Automation workflows, see AWS Systems Manager Automation (p. 278).
- AWS Lambda functions
  For more information about Lambda functions, see Working with Lambda Functions in the AWS Lambda Developer Guide.
- AWS Step Functions tasks
  For more information about Step Functions, see the AWS Step Functions Developer Guide.

This means you can use maintenance windows to perform tasks like the following on your selected targets:

- Install or update applications.
- Apply patches.
- Install or update SSM Agent.
- Run PowerShell commands and Linux shell scripts by using a Systems Manager Run Command task.
- Build Amazon Machine Images (AMIs), boot-strap software, and configure instances by using a Systems Manager Automation task.
• Run AWS Lambda functions that trigger additional actions, such as scanning your instances for patch updates.
• Run AWS Step Functions state machines to perform tasks such as removing an instance from an Elastic Load Balancing environment, patching the instance, and then adding the instance back to the Elastic Load Balancing environment.
• Target instances that are offline by specifying an AWS resource group as the target.

Contents
• Controlling access to maintenance windows (p. 621)
• Working with maintenance windows (console) (p. 634)
• Systems Manager Maintenance Windows tutorials (AWS CLI) (p. 640)
• Maintenance window walkthroughs (p. 685)
• Reference: Maintenance window scheduling and active period options (p. 693)

Controlling access to maintenance windows

Before users in your account can create and schedule maintenance window tasks, they must be granted the necessary permissions. To grant these permissions to users, an administrator must perform these two tasks:

Task 1: Configure instance permissions

Provide the Maintenance Windows service with the AWS Identity and Access Management (IAM) permissions needed to run maintenance window tasks on your instances by doing one of the following:

• Create a custom service role for maintenance window tasks
• Create a service-linked role for Systems Manager

You specify one of these roles as part of the configuration when you create a maintenance window task. This allows Systems Manager to run tasks in maintenance windows on your behalf.

Note
A service-linked role for Systems Manager might already have been created in your account. Currently, the service-linked role also provides permissions for the Inventory capability.

To help you decide whether to use a custom service role or the Systems Manager service-linked role with a maintenance window task, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).

Task 2: Configure user permissions

Granting iam:PassRole permissions to the users in your account who assigns tasks to maintenance windows. This allows them to pass the role to the maintenance window service. Without this explicit permission, a user can't assign tasks to a maintenance window.

Before you begin

In order to complete the tasks in the section, you need one or both of the following resources.

• You are assigning permissions to IAM users or groups. These users or groups should already have been granted general permissions for working with maintenance windows. This can be done by assigning the IAM policy AmazonSSMFullAccess to the users or groups, or by creating and assigning an IAM policy that provides a smaller set of access permissions for Systems Manager that covers maintenance
window tasks. For more information, see Create user groups (p. 27) and Create users and assign
permissions (p. 28).

• (Optional) For maintenance windows that run Run Command tasks, you can choose for Amazon Simple
Notification Service (Amazon SNS) status notifications to be sent. For information about configuring
Amazon SNS notifications for Systems Manager, including information about creating an IAM role to
use for sending SNS notifications, see Monitoring Systems Manager status changes using Amazon SNS
notifications (p. 1151).

Should I use a service-linked role or a custom service role to run
maintenance window tasks?

To run maintenance tasks on your target instances, the Maintenance Windows service must have
permission to access and run tasks on your instances. You can provide this permission by specifying
either the Systems Manager service-linked role or a custom service role as part of a task configuration.

The type of role you should choose depends on the following factors:

**Custom service role:** Use a custom service role for maintenance window tasks in these cases:

• If you want to use a more restrictive set of permissions than those provided by the service-linked role.
The service-linked role supports very limited resource-level constraints. For example, say you want
to allow maintenance window tasks to run on a limited set of instances, or you want to allow only
certain SSM documents run on your target instances. In these cases, you specify stricter permissions in
a custom service role.

• If you need a more permissive or expanded set of permissions than those provided by the service-
linked role. Some actions in Automation documents require expanded permissions.

For example, some Automation actions work with AWS CloudFormation stacks. Therefore, the
permissions `cloudformation:CreateStack`, `cloudformation:DescribeStack`, and
`cloudformation:DeleteStack` are required.

Another example: the Automation document `AWS-CopySnapshot` requires permission to create an
Amazon Elastic Block Store (Amazon EBS) snapshot, and so the service role needs the permission
`ec2:CreateSnapshot`. This permission isn't included in the service-linked role for Systems Manager.

For information about the role permissions needed by Automation documents, see the document

**Systems Manager service-linked role:** We recommend that you use a Systems Manager service-linked
role in all other cases.

For more information about the Systems Manager service-linked role, see Using service-linked roles for
Systems Manager (p. 1122).

**Topics**

• Control access to maintenance windows (console) (p. 622)
• Control access to maintenance windows (AWS CLI) (p. 626)
• Control access to maintenance windows (Tools for Windows PowerShell) (p. 630)
• Troubleshooting IAM maintenance window permissions (p. 633)

**Control access to maintenance windows (console)**

The following procedures describe how to use the AWS Systems Manager console to create the required
roles and permissions for maintenance windows.
Task 1: (Optional) Create a custom service role for maintenance windows (console)

Use the following procedure to create a custom service role for the Maintenance Windows capability so that Systems Manager can run tasks on your behalf.

Important
A custom service role is not required if you choose to use a Systems Manager service-linked role to let maintenance windows run tasks on your behalf instead. If you do not have a Systems Manager service-linked role in your account, you can create it when you create or update a maintenance window task using the Systems Manager console. For more information, see the following topics:

- Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622)
- Using service-linked roles for Systems Manager (p. 1122)
- Assign tasks to a maintenance window (console) (p. 637)

To create a custom service role (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
3. Mark the following selections:
   1. Select type of trusted entity area: AWS service
   2. Choose the service that will use this role area: Systems Manager
4. Choose Next: Permissions.
5. In the list of policies, select the box next to AmazonSSMMaintenanceWindowRole, and then choose Next: Tags.
6. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose Next: Review.
7. In Role name, enter a name that identifies this role as a Maintenance Windows role; for example my-maintenance-window-role.
8. (Optional) Change the default role description to reflect the purpose of this role. For example: Performs maintenance window tasks on your behalf.
9. Choose Create role. The system returns you to the Roles page.
10. Choose the name of the role you just created.
11. Choose the Trust relationships tab, and then choose Edit trust relationship.
12. Verify that the following policy appears in the Policy Document field:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      }
    }
  ]
}
```
13. Choose **Update Trust Policy**, and then copy or make a note of the role name and the **Role ARN** value on the **Summary** page. You specify this information when you create your maintenance window.

14. (Optional) If you plan to configure a maintenance window to send notifications about command statuses using Amazon SNS, when run through a Run Command command task, do the following:

   1. Choose the **Permissions** tab.
   2. Choose **Add inline policy**, and then choose the **JSON** tab.
   3. In **Policy Document**, paste the following:

       ```json
       {
         "Version": "2012-10-17",
         "Statement": [
           {
             "Effect": "Allow",
             "Action": "iam:PassRole",
             "Resource": "sns-access-role-arn"
           }
         ]
       }
       ```

       Replace **sns-access-role-arn** with the ARN of the existing IAM role to use to send Amazon Simple Notification Service (Amazon SNS) notifications related to the maintenance window, in the format of `arn:aws:iam::<account-id>:role/role-name`. For example: `arn:aws:iam::123456789012:role/my-sns-access-role`. For information about configuring Amazon SNS notifications for Systems Manager, including information about creating an IAM role to use for sending SNS notifications, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

       **Note**

       In the Systems Manager console, this ARN is selected in the **IAM Role** list on the **Register run command task** page. For information, see Assign tasks to a maintenance window (console) (p. 637). In the Systems Manager API, this ARN is entered as the value of `ServiceRoleArn` in the `SendCommand` request.

   4. Choose **Review policy**.
   5. For **Name**, enter a name to identify this as a policy to allow sending Amazon SNS notifications.
   15. Choose **Create policy**.

### Task 2: Assign the IAM PassRole policy to an IAM user or group (console)

When you register a task with a maintenance window, you specify either a custom service role or a Systems Manager service-linked role to run the actual task operations. This is the role that the service assumes 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. For information, see Granting a User Permissions to Pass a Role to an AWS Service in the **IAM User Guide**.

Depending on whether you are assigning the `iam:Passrole` permission to an individual user or a group, use one of the following procedures to provide the minimum permissions required to register tasks with a maintenance window.
To assign the IAM PassRole policy to an IAM user account (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose Users, and then choose the name of the user account you want to update.
3. On the Permissions tabs, in the policies list, verify that the AmazonSSMFullAccess policy is listed, or that there is a comparable policy that gives the IAM user permission to call the Systems Manager API. Add the permission if it is not included already. For information, see Adding and Removing IAM Policies (Console) in the IAM User Guide.
4. Choose Add inline policy, and then choose the JSON tab.
5. In Policy Document, paste the following:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "custom-role-arn"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ListRoles",
      "Resource": "arn:aws:iam::account-id:role/"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ListRoles",
      "Resource": "arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/"
    }
  ]
}
```

Replace `custom-role-arn` with the ARN of the custom maintenance window role you created earlier, such as arn:aws:iam::123456789012:role/my-maintenance-window-role.

Replace `account-id` in the two iam:ListRoles permissions with the ID of your AWS account. Adding this permission for the resource arn:aws:iam::account-id:role/ allows a user to view and choose from customer roles in the console when they create a maintenance window task. Adding this permission for arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/ allows a user to choose the Systems Manager service-linked role in the console when they create a maintenance window task.

7. On the Review policy page, enter a name in the Name box to identify this PassRole policy, such as `my-iam-passrole-policy`, and then choose Create policy.

To assign the IAM PassRole policy to an IAM group (console)

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Groups.
3. In the list of groups, select the name of the group you want to assign the iam:PassRole permission to.
4. On the Permissions tab, in the Inline Policies section, do one of the following:
   - If no inline policies have been added yet, choose click here.
   - If one or more inline policies have been added, choose Create Group Policy.
5. Select **Custom Policy**, and then choose **Select**.
6. For **Policy Name**, enter a name to identify this as a maintenance windows PassRole policy for your group, such as `my-group-iam-passrole-policy`.
7. In **Policy Document**, paste the following:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": "iam:PassRole",
         "Resource": "custom-role-arn"
       },
       {
         "Effect": "Allow",
         "Action": "iam:ListRoles",
         "Resource": "arn:aws:iam::account-id:role/
       },
       {
         "Effect": "Allow",
         "Action": "iam:ListRoles",
         "Resource": "arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/"
       }
     ]
   }
   ``

   Replace `custom-role-arn` with the ARN of the custom maintenance window role you created earlier, such as `arn:aws:iam::123456789012:role/my-maintenance-window-role`.

   Replace `account-id` in the two `iam:ListRoles` permissions with the ID of your AWS account. Adding this permission for the resource `arn:aws:iam::account-id:role/` allows users in the group to view and choose from customer roles in the console when they create a maintenance window task. Adding this permission for `arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/` allows users in the group to choose the Systems Manager service-linked role in the console when they create a maintenance window task.

8. Choose **Apply Policy**.

### Control access to maintenance windows (AWS CLI)

The following procedures describe how to use the AWS CLI to create the required roles and permissions for Maintenance Windows.

**Topics**

- Task 1: (Optional) Create a custom service role for maintenance windows (AWS CLI) (p. 626)
- Task 2: Assign the IAM PassRole policy to an IAM user or group (p. 628)

**Task 1: (Optional) Create a custom service role for maintenance windows (AWS CLI)**

**Important**

A custom service role is not required if you choose to use a Systems Manager service-linked role to let maintenance windows run tasks on your behalf instead. If you do not have a Systems Manager service-linked role in your account, you can create it when you create or update a maintenance window task using the Systems Manager console. For more information, see the following topics:
• Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622)
• Using service-linked roles for Systems Manager (p. 1122)
• Assign tasks to a maintenance window (console) (p. 637)

1. Copy and paste the following trust policy into a text file. Save the file with the following name and file extension: mw-role-trust-policy.json.

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

2. Open the AWS CLI and run the following command in the directory where you placed mw-role-trust-policy.json in order to create a maintenance window role called my-maintenance-window-role. The command assigns the policy you created in the previous step to this role.

**Linux**

```bash
aws iam create-role
    --role-name "my-maintenance-window-role"
    --assume-role-policy-document file://mw-role-trust-policy.json
```

**Windows**

```bash
aws iam create-role
    --role-name "my-maintenance-window-role"
    --assume-role-policy-document file://mw-role-trust-policy.json
```

The system returns information like the following:

```json
{
    "Role": {
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "ssm.amazonaws.com"
                    }
                }
            ]
        },
        "RoleId": "AROAIIZKPSKSS2EXAMPLE",
        "CreateDate": "2017-04-04T03:40:17.373Z",
        "RoleName": "my-maintenance-window-role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/my-maintenance-window-role"
    }
}
```
Controlling access

Note

Make a note of the RoleName and the Arn values. You specify these when you create a maintenance window that uses this custom role.

3. Run the following command to attach the AmazonSSMMaintenanceWindowRole managed policy to the role you created in step 2.

Linux

aws iam attach-role-policy  
--role-name "my-maintenance-window-role"  
--policy-arn "arn:aws:iam::aws:policy/service-role/AmazonSSMMaintenanceWindowRole"

Windows

aws iam attach-role-policy ^  
--role-name "my-maintenance-window-role" ^  
--policy-arn "arn:aws:iam::aws:policy/service-role/AmazonSSMMaintenanceWindowRole"

Task 2: Assign the IAM PassRole policy to an IAM user or group

When you register a task with a maintenance window, you specify either a custom service role or a Systems Manager service-linked role to run the actual task operations. This is the role that the service assumes 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. For information, see Granting a User Permissions to Pass a Role to an AWS Service in the IAM User Guide.

To assign the IAM PassRole policy to an IAM user account or group (AWS CLI)

1. Copy and paste the following IAM policy into a text editor and save it with the following name and file extension: mw-passrole-policy.json.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": "iam:PassRole",
         "Resource": "custom-role-arn"
      },
      {
         "Effect": "Allow",
         "Action": "iam:ListRoles",
         "Resource": "arn:aws:iam::account-id:role/"
      },
      {
         "Effect": "Allow",
         "Action": "iam:ListRoles",
         "Resource": "arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/"
      }
   ]
}
```
Replace `custom-role-arn` with the ARN of the custom maintenance window role you created earlier, such as `arn:aws:iam::123456789012:role/my-maintenance-window-role`.

Replace `account-id` in the two `iam:ListRoles` permissions with the ID of your AWS account. Adding this permission for the resource `arn:aws:iam::account-id:role/` allows users in the group to view and choose from customer roles in the console when they create a maintenance window task. Adding this permission for `arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/` allows users in the group to choose the Systems Manager service-linked role in the console when they create a maintenance window task.

2. Open the AWS CLI.
3. Depending on whether you are assigning the permission to an IAM user or group, run one of the following commands.

   - **For an IAM user**:

     **Linux**
     ```bash
     aws iam put-user-policy \
     --user-name "user-name" \
     --policy-name "policy-name" \
     --policy-document file://path-to-document
     ```

     **Windows**
     ```bash
     aws iam put-user-policy ^
     --user-name "user-name" ^
     --policy-name "policy-name" ^
     --policy-document file://path-to-document
     ```

     For `user-name`, specify the IAM user who assigns tasks to maintenance windows. For `policy-name`, specify the name you want to use to identify the policy, such as `my-iam-passrole-policy`. For `path-to-document`, specify the path to the file you saved in step 1. For example: `file://C:\Temp\mw-passrole-policy.json`

   - **Note**
     To grant access for a user to register tasks for maintenance windows using the AWS Systems Manager console, you must also assign the `AmazonSSMFullAccess` policy to your user account (or an IAM policy that provides a smaller set of access permissions for Systems Manager that covers maintenance window tasks. For more information, see Create user groups (p. 27) and Create users and assign permissions (p. 28). Run the following command to assign the `AmazonSSMFullAccess` policy to your account:

     **Linux**
     ```bash
     aws iam attach-user-policy \
     --policy-arn "arn:aws:iam::aws:policy/AmazonSSMFullAccess" \
     --user-name "user-name"
     ```

     **Windows**
     ```bash
     aws iam attach-user-policy ^
     --policy-arn "arn:aws:iam::aws:policy/AmazonSSMFullAccess" ^
     --user-name "user-name"
     ```

   - **For an IAM group**:
Controlling access

4. Run the following command to verify that the policy has been assigned to the group:

Linux

```
aws iam list-group-policies \\
--group-name "group-name"
```

Windows

```
aws iam list-group-policies ^
--group-name "group-name" ^
```

---

Control access to maintenance windows (Tools for Windows PowerShell)

The following procedures describe how to use the Tools for Windows PowerShell to create the required roles and permissions for the Maintenance Windows capability.

Topics
Task 1: (Optional) Create a custom service role for maintenance windows (Tools for Windows PowerShell) (p. 631)

Task 2: Assign the IAM PassRole policy to an IAM user or group (PowerShell) (p. 632)

Task 1: (Optional) Create a custom service role for maintenance windows (Tools for Windows PowerShell)

Important
A custom service role is not required if you choose to use a Systems Manager service-linked role to let maintenance windows run tasks on your behalf instead. If you do not have a Systems Manager service-linked role in your account, you can create it when you create or update a maintenance window task using the Systems Manager console. For more information, see the following topics:

- Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622)
- Using service-linked roles for Systems Manager (p. 1122)
- Assign tasks to a maintenance window (console) (p. 637)

1. Copy and paste the following trust policy into a text file. Save the file with the following name and file extension: mw-role-trust-policy.json.

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

2. Open Tools for Windows PowerShell and run the following command to create a role with a name that identifies this role as a maintenance window role; for example my-maintenance-window-role. The role uses the policy that you created in the previous step:

   ```powershell
   New-IAMRole 
   -RoleName "my-maintenance-window-role" 
   -AssumeRolePolicyDocument (Get-Content -raw .\mw-role-trust-policy.json)
   ```

   The system returns information like the following.

   ```text
   Arn : arn:aws:iam::123456789012:role/mw-task-role
   AssumeRolePolicyDocument : ExampleDoc12345678
   CreateDate : 4/4/2017 11:24:43
   Path : /
   RoleId : AROAIIZKBKS2LEXAMPLE
   RoleName : my-maintenance-window-role
   ```

3. Run the following command to attach the AmazonSSMMaintenanceWindowRole managed policy to the role you created in the previous step:

   ```powershell
   Register-IAMRolePolicy 
   ```
Task 2: Assign the IAM PassRole policy to an IAM user or group (PowerShell)

When you register a task with a maintenance window, you specify either a custom service role or a Systems Manager service-linked role to run the actual task operations. This is the role that the service assumes 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. For information, see Granting a User Permissions to Pass a Role to an AWS Service in the IAM User Guide.

1. Copy and paste the following IAM policy into a text editor and save it with the following name and file extension: mw-passrole-policy.json.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "custom-role-arn"
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::account-id:role/
        },
        {
            "Effect": "Allow",
            "Action": "iam:ListRoles",
            "Resource": "arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/"
        }
    ]
}
```

Replace `custom-role-arn` with the ARN of the custom maintenance window role you created earlier, such as `arn:aws:iam::123456789012:role/my-maintenance-window-role`.

Replace `account-id` in the two `iam:ListRoles` permissions with the ID of your AWS account. Adding this permission for the resource `arn:aws:iam::account-id:role/` allows users in the group to view and choose from customer roles in the console when they create a maintenance window task. Adding this permission for `arn:aws:iam::account-id:role/aws-service-role/ssm.amazonaws.com/` allows users in the group to choose the Systems Manager service-linked role in the console when they create a maintenance window task.


3. Depending on whether you are assigning the permission to an IAM user or group, run one of the following commands.

   - For an IAM user:

     ```powershell
     Write-IAMUserPolicy `n-UserName "user-name" `n-PolicyDocument (Get-Content -raw path-to-document) `n-PolicyName "policy-name"
     ```
For **user-name**, specify the IAM user who assigns tasks to maintenance windows. For **policy-name**, specify the name you want to use to identify the policy, such as **my-iam-passrole-policy**. For **path-to-document**, specify the path to the file you saved in step 1. For example: C:\temp\passrole-policy.json

**Note**
If you plan to register tasks for maintenance windows using the AWS Systems Manager console, you must also assign the AmazonSSMFullAccess policy to your user account. Run the following command to assign this policy to your account:

```bash
Register-IAMUserPolicy
-UserName "user-name"
-PolicyArn "arn:aws:iam::aws:policy/AmazonSSMFullAccess"
```

• **For an IAM group:**

```bash
Write-IAMGroupPolicy
-GroupName "group-name"
-PolicyDocument (Get-Content -raw path-to-document)
-PolicyName "policy-name"
```

For **group-name**, specify the IAM group that assigns tasks to maintenance windows. For **policy-name**, specify the name you want to use to identify the policy, such as **my-iam-passrole-policy**. For **path-to-document**, specify the path to the file you saved in step 1. For example: C:\temp\passrole-policy.json

**Note**
If you plan to register tasks for maintenance windows using the AWS Systems Manager console, you must also assign the AmazonSSMFullAccess policy to your user account. Run the following command to assign this policy to your group:

```bash
Register-IAMGroupPolicy
-GroupName "group-name"
-PolicyArn "arn:aws:iam::aws:policy/AmazonSSMFullAccess"
```

4. Run the following command to verify that the policy has been assigned to the group:

```bash
Get-IAMGroupPolicies
-GroupName "group-name"
```

### Troubleshooting IAM maintenance window permissions

Use the following information to help you troubleshoot common issues with Maintenance Windows permissions in AWS Systems Manager.

**Edit task error:** On the page for editing a maintenance window task, the IAM role list returns an error message: "We couldn't find the IAM maintenance window role specified for this task. It might have been deleted, or it might not have been created yet."

**Problem 1:** The IAM maintenance window role you originally specified was deleted after you created the task.
Possible fixes: (1) Select a different IAM maintenance window role, if one exists in your account, or create a new one and select it for the task. (2) Create or select a Systems Manager service-linked role. For more information, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).

Problem 2: If the task was created using the AWS CLI, Tools for Windows PowerShell, or an AWS SDK, a non-existent IAM maintenance window role name could have been specified. For example, the IAM maintenance window role could have been deleted before you created the task, or the role name could have been typed incorrectly, such as myrole instead of my-role.

Possible fixes: (1) Select the correct name of the IAM maintenance window role you want to use, or create a new one to specify for the task. (2) Create or select a Systems Manager service-linked role. For more information, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).

Working with maintenance windows (console)

This section describes how to create, configure, and update, and delete maintenance windows using the Systems Manager console. This section also provides information about managing the targets and tasks of a maintenance window.

Important

We recommend that you initially create and configure maintenance windows in a test environment.

Before You Begin

Before you create a maintenance window, you must configure access to Maintenance Windows. For more information, see Controlling access to maintenance windows (p. 621).

Topics

- Create a maintenance window (console) (p. 634)
- Assign targets to a maintenance window (console) (p. 635)
- Assign tasks to a maintenance window (console) (p. 637)
- Update or delete a maintenance window (console) (p. 639)

Create a maintenance window (console)

In this procedure, you create a maintenance window and specify its basic options, such as name, schedule, and duration. In later steps, you choose the targets, or resources, that it updates and the tasks that run during the maintenance window execution.

Note

For an explanation of how the various schedule-related options for maintenance windows relate to one another, see Reference: Maintenance window scheduling and active period options (p. 693).

To create a maintenance window (console)

2. In the navigation pane, choose Maintenance Windows.
3. Choose Create a maintenance window.
4. For Name, enter a descriptive name to help you identify this maintenance window as a test maintenance window.
5. For Description, enter a description.
6. Choose Allow unregistered targets if you want to allow a maintenance window task to run on managed instances, even if you have not registered those instances as targets. If you choose this
option, then you can choose the unregistered instances (by instance ID) when you register a task with the maintenance window.

If you don't choose this option, then you must choose previously-registered targets when you register a task with the maintenance window.

7. Specify a schedule for the maintenance window by using one of the three scheduling options.

For information about building cron/rate expressions, see Reference: Cron and rate expressions for Systems Manager (p. 1189).

8. For Duration, enter the number of hours the maintenance window will run. The value you specify determines the specific end time for the maintenance window based on the time it begins. No maintenance window tasks are permitted to start after the resulting endtime minus the number of hours you specify for Stop initiating tasks in the next step.

For example, if the maintenance window starts at 3 PM, the duration is three hours, and the Stop initiating tasks value is one hour, no maintenance window tasks can start after 5 PM.

9. For Stop initiating tasks, enter the number of hours before the end of the maintenance window that the system should stop scheduling new tasks to run.

10. (Optional) For Start date (optional), specify a date and time, in ISO-8601 Extended format, for when you want the maintenance window to become active. This allows you to delay activation of the maintenance window until the specified future date.

11. (Optional) For End date (optional), specify a date and time, in ISO-8601 Extended format, for when you want the maintenance window to become inactive. This allows you to set a date and time in the future after which the maintenance window no longer runs.

12. (Optional) For Time zone (optional), specify the time zone to base scheduled maintenance window executions on, in Internet Assigned Numbers Authority (IANA) format. For example: "America/Los_Angeles", "etc/UTC", or "Asia/Seoul".

For more information about valid formats, see the Time Zone Database on the IANA website.

13. (Optional) In the Manage tags area, apply one or more tag key name/value pairs to the maintenance window.

Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a maintenance window to identify the type of tasks it runs, the types of targets, and the environment it runs in. In this case, you could specify the following key name/value pairs:

- Key=TaskType,Value=AgentUpdate
- Key=OS,Value=Windows
- Key=Environment,Value=Production

14. Choose Create maintenance window. The system returns you to the maintenance window page. The state of the maintenance window you just created is Enabled.

Assign targets to a maintenance window (console)

In this procedure, you register a target with a maintenance window. In other words, you specify which resources the maintenance window performs actions on.

To assign targets to a maintenance window (console)

1. In the list of maintenance windows, choose the maintenance window to add targets to.

2. Choose Actions, and then choose Register targets.

3. (Optional) For Target name, enter a name for the targets.

4. (Optional) For Description, enter a description.
5. (Optional) For **Owner information**, specify information to include in any CloudWatch Events raised while running tasks for these targets in this maintenance window.

   For information about using CloudWatch Events to monitor Systems Manager events, see [Monitoring Systems Manager events with Amazon CloudWatch Events](p. 1147).

6. In the **Targets** area, choose one of the options described in the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Specify instance tags**  | **For Instance tags**, specify one or more tag keys and (optional) values that have been or will be added to managed instances in your account. When the maintenance window runs, it attempts to perform tasks on all of the managed instances to which these tags have been added.  
  If you specify more than one tag key, an instance must be tagged with all the tag keys and values you specify to be included in the target group. |
| **Choose instances manually** | From the list, select the box for each instance that you want to include in the maintenance window target.                                        
  The list includes all instances in your account that are configured for use with Systems Manager.  
  If you don't see an instance you want to include in the target, verify that the required setup steps have been completed:  
  • For EC2 instances, see [Setting up AWS Systems Manager](p. 25).  
  • For on-premises instances and virtual machines (VMs), see [Setting up AWS Systems Manager for hybrid environments](p. 43) |
| **Choose resource groups** | **For Resource group**, choose the name of an existing resource group in your account from the list.                                         
  For information about creating and working with resource groups, see the following topics:  
  • [Resource Groups and Tagging for AWS](AWS News Blog) in the *AWS News Blog*  
  **For Resource types**, select up to five available resource types, or choose **All resource types**.  
  If the tasks you assign to the maintenance window do not act on one of the resource types |
### Assign tasks to a maintenance window (console)

In this procedure, you add a task to a maintenance window. Tasks are the actions performed on a resource during a maintenance window execution.

The following four types of tasks can be added to a maintenance window:

- Systems Manager Run Command commands
- Systems Manager Automation workflows
- AWS Lambda functions
- AWS Step Functions tasks

#### To assign tasks to a maintenance window

1. In the list of maintenance windows, choose a maintenance window.
2. Choose Actions, and then choose the option for the type of task you want to register with the maintenance window:
   - Register Run command task
   - Register Automation task
   - Register Lambda task
   - Register Step Functions task
3. For Name, enter a name for the task.
4. For Description, enter a description.
5. For Document, choose the SSM Command or Automation document that defines the tasks to run.
6. For Document version (for Automation tasks), choose the document version to use.
7. For Task priority, specify a priority for this task. Zero (0) is the highest priority. Tasks in a maintenance window are scheduled in priority order with tasks that have the same priority scheduled in parallel.
8. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

Note
If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.

9. (Optional) For Rate control:

• For Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.

Note
If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

• For Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

10. In the IAM service role area, choose one of the following options to provide permissions for Systems Manager to run tasks on your target instances:

• Create and use a service-linked role for Systems Manager

Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role. Additionally, AWS automatically defines and sets the permissions of service-linked roles, depending on the actions that the linked service performs on your behalf.

Note
If a service-linked role has already been created for your account, choose Use the service-linked role for Systems Manager.

• Use a custom service role

You can create a custom service role for maintenance window tasks if you want to use stricter permissions than those provided by the service-linked role.

If you need to create a custom service role, see one of the following topics:

• Control access to maintenance windows (console) (p. 622)
• Control access to maintenance windows (AWS CLI) (p. 626)
• Control access to maintenance windows (Tools for Windows PowerShell) (p. 630)

To help you decide whether to use a custom service role or the Systems Manager service-linked role with a maintenance window task, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).

11. In the Input Parameters section, specify parameters for the document. For Automation documents, the system auto-populates some of the values. You can keep or replace these values.

12. Complete the wizard.
Update or delete a maintenance window (console)

You can update or delete a maintenance window. You can also update or delete the targets or tasks of a maintenance window. If you edit the details of a maintenance window, you can change the schedule, targets, and tasks. You can also specify names and descriptions for windows, targets, and tasks, which helps you better understand their purpose, and makes it easier to manage your queue of windows.

This section describes how to update or delete a maintenance window, targets, and tasks by using the AWS Systems Manager console. For examples of how to do this by using the AWS CLI, see Tutorial: Update a maintenance window (AWS CLI) (p. 676).

**Topics**

- Update or delete a maintenance window (console) (p. 639)
- Update or delete maintenance window targets (console) (p. 639)
- Update or delete maintenance window tasks (console) (p. 640)

Update or delete a maintenance window (console)

You can update a maintenance window to change its name, description, and schedule, and whether the maintenance window should allow unregistered targets.

**To update or delete a maintenance window**

2. In the navigation pane, choose Maintenance Windows.
3. Choose the maintenance window that you want to update or delete, and then do one of the following:
   - Choose **Delete**. The system prompts you to confirm your actions.
   - Choose **Edit**. On the Edit maintenance window page, change the values and options that you want, and then choose Edit maintenance window.

   For information about the configuration choices you can make, see Create a maintenance window (console) (p. 634).

Update or delete maintenance window targets (console)

You can update or delete the targets of a maintenance window. If you choose to update a maintenance window target you can specify a new target name, description, and owner. You can also choose different targets.

**To update or delete the targets of a maintenance window**

2. In the navigation pane, choose Maintenance Windows.
3. Choose the name of the maintenance window that you want to update, and then do one of the following:
   - To update targets, choose **Edit**.
   - To delete targets, choose Deregister targets, and then choose the **Targets** tab.

   Choose the target to delete, and then choose **Deregister target**. In the **Deregister maintenance windows target** window, leave the **Safely deregister target** option selected if you want the system to check if the target is referenced by any tasks before deleting it. If the target is referenced by a task, the system returns an error and doesn't delete the target. Clear the **Safely
deregister target option if you want the system to delete the target even if it is referenced by a task.

Choose Deregister.

Update or delete maintenance window tasks (console)

You can update or delete the tasks of a maintenance window. If you choose to update, you can specify a new task name, description, and owner. For Run Command and Automation tasks, you can choose a different SSM document for the tasks. You can't, however, edit a task to change its type. For example, if you created an Automation task, you can't edit that task and change it to a Run Command task.

To update or delete the tasks of a maintenance window (console)

2. In the navigation pane, choose Maintenance Windows.
3. Choose the maintenance window that you want to update.
4. Choose the Tasks tab.
5. If you want to delete a task, choose the small x button next to Edit. If you want to edit the task, choose Edit.
6. Change the values and options that you want, and then choose Edit Task. The system returns you to the maintenance window page.

Systems Manager Maintenance Windows tutorials (AWS CLI)

This section includes tutorials that help you learn how to use the AWS Command Line Interface (AWS CLI) to do the following:

• Create and configure a maintenance window
• View information about a maintenance window
• View information about maintenance windows tasks and task executions
• Update a maintenance window
• Delete a maintenance window

Complete prerequisites

Before trying these tutorials, complete the following prerequisites.

• Configure the AWS CLI on your local machine: Before you can run AWS CLI commands, you must install and configure the CLI on your local machine. For information, see Install or upgrade AWS command line tools (p. 57).
• Verify maintenance window roles and permissions: An AWS administrator in your account must grant you the AWS Identity and Access Management (IAM) permissions you need to manage maintenance windows using the CLI. For information, see Controlling access to maintenance windows (p. 621).
• Create or configure a Systems Manager-compatible instance: You need at least one EC2 instance that is configured for use with Systems Manager in order to complete the tutorials. This means that SSM Agent is installed on the instance, and an IAM instance profile for Systems Manager is attached to the instance.

We recommend launching an instance from one of the following Amazon Machine Image (AMI) types. SSM Agent is preinstalled on each one:
• Windows Server 2008-2012 R2 AMIs published in November 2016 or later
• Windows Server 2016 and 2019
• Amazon Linux
• Amazon Linux 2
• Ubuntu Server 16.04
• Ubuntu Server 18.04
• Amazon ECS-Optimized

For information about installing SSM Agent on an instance, see the following topics:
• Installing and configuring SSM Agent on Windows Server instances (p. 65)
• Installing and configuring SSM Agent on EC2 instances for Linux (p. 69)

For information about creating and attaching an IAM instance profile for Systems Manager to your instance, see the following topics:
• Create an IAM instance profile for Systems Manager (p. 30)
• Attach an IAM instance profile to an EC2 instance (p. 35)

• **Create additional resources as needed**: Many Run Command tasks do not require you to create resources other than those listed in this prerequisites topic. For that reason, we provide a simple Run Command task for you to use your first time through the tutorials. You also need an EC2 instance that is configured to use with Systems Manager, as described above. After you configure that instance, you can register a simple Run Command task.

The Systems Manager Maintenance Windows capability supports running four types of tasks:
• Systems Manager Run Command commands
• Systems Manager Automation workflows
• AWS Lambda functions
• AWS Step Functions tasks

In general, if a maintenance window task that you want to run requires additional resources, you should create them first. For example, if you want a maintenance window that runs an AWS Lambda function, create the Lambda function before you begin; for a Run Command task, create the S3 bucket that you can save command output to (if you plan to do so); and so on.

**Keep track of resource IDs**

As you complete the tasks in this AWS CLI tutorial, keep track of resource IDs generated by the commands you run. You use many of these as input for subsequent commands. For example, when you create the maintenance window, the system provides you with a maintenance window ID in this format:

```json
{
  "WindowId": "mw-0c50858d01EXAMPLE"
}
```

Make a note of the following system-generated IDs because the tutorials in this section use them:

• WindowId
• WindowTargetId
• WindowTaskId
• WindowExecutionId
• TaskExecutionId
• InvocationId
• ExecutionId

You also need the ID of the EC2 instance you plan to use in the tutorial. For example: i-02573cafcfEXAMPLE

Tutorials
• Tutorial: Create and configure a maintenance window (AWS CLI) (p. 642)
• Tutorial: View information about maintenance windows (AWS CLI) (p. 663)
• Tutorial: View information about tasks and task executions (AWS CLI) (p. 674)
• Tutorial: Update a maintenance window (AWS CLI) (p. 676)
• Tutorial: Delete a maintenance window (AWS CLI) (p. 684)

Tutorial: Create and configure a maintenance window (AWS CLI)

This tutorial demonstrates how to use the AWS CLI to create and configure a maintenance window, its targets, and its tasks. The main path through the tutorial consists of simple steps. You create a single maintenance window, identify a single target, and set up a simple task for the maintenance window to run. Along the way, we provide information you can use to try more complicated scenarios.

As you follow the steps in this tutorial, replace the values in italicized red text with your own options and IDs. For example, replace the maintenance window ID mw-0c50858d01EXAMPLE and the instance ID i-02573cafcfEXAMPLE with IDs of resources you create.

Contents
• Step 1: Create the maintenance window (AWS CLI) (p. 642)
• Step 2: Register a target instance with the maintenance window (AWS CLI) (p. 643)
• Step 3: Register a task with the maintenance window (AWS CLI) (p. 648)

Step 1: Create the maintenance window (AWS CLI)

In this step, you create a maintenance window and specify its basic options, such as name, schedule, and duration. In later steps, you choose the instance it updates and the task it runs.

In our example, you create a maintenance window that runs every five minutes.Normally, you would not run a maintenance window this frequently. However, this rate lets you see your tutorial results quickly. We'll show you how to change to a less frequent rate after the task has run successfully.

To create a maintenance window (AWS CLI)

1. Open the AWS CLI and run the following command on your local machine to create a maintenance window that does the following:
   • Runs every five minutes for up to two hours (as needed).
   • Prevents new tasks from starting within one hour of the end of the maintenance window execution.
   • Allows unassociated targets (instances that you haven't registered with the maintenance window).
   • Indicates through the use of custom tags that its creator intends to use it in a tutorial.

   Linux

   ```bash
   aws ssm create-maintenance-window \\
   --name "My-First-Maintenance-Window" \\
   --schedule "rate(5 minutes)"
   ```
Windows

```bash
aws ssm create-maintenance-window
  --name "My-First-Maintenance-Window"
  --schedule "rate(5 minutes)"
  --duration 2 --cutoff 1
  --allow-unassociated-targets
  --tags "Key=Purpose,Value=Tutorial"
```

The system returns information like the following:

```
{
  "WindowId": "mw-0c50858d01EXAMPLE"
}
```

2. Now run this command to view details about this and any other maintenance windows already in your account:

```bash
aws ssm describe-maintenance-windows
```

The system returns information like the following:

```
{
  "WindowIdentities": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "Name": "My-First-Maintenance-Window",
      "Enabled": true,
      "Duration": 2,
      "Cutoff": 1,
    }
  ]
}
```

Continue to Step 2: Register a target instance with the maintenance window (AWS CLI) (p. 643).

**Step 2: Register a target instance with the maintenance window (AWS CLI)**

In this step, you register a target with your new maintenance window. In this case, you specify which instance to update when the maintenance window runs.

For an example of registering more than one instance at a time using instance IDs, examples of using tags to identify multiple instances, and examples of specifying resource groups as targets, see Examples: Register targets with a maintenance window (p. 645).

**Note**
You should already have created an EC2 instance to use in this step, as described in the Maintenance Windows tutorial prerequisites (p. 640).

**To register a target instance with a maintenance window (AWS CLI)**

1. Run the following command on your local machine:
Linux

```bash
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--resource-type "INSTANCE" \
--target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE"
```

Windows

```bash
aws ssm register-target-with-maintenance-window ^
--window-id "mw-0c50858d01EXAMPLE" ^
--resource-type "INSTANCE" ^
--target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE"
```

The system returns information like the following:

```
{
  "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
}
```

2. Now run the following command on your local machine to view details about your maintenance window target:

Linux

```bash
aws ssm describe-maintenance-window-targets \
--window-id "mw-0c50858d01EXAMPLE"
```

Windows

```bash
aws ssm describe-maintenance-window-targets ^
--window-id "mw-0c50858d01EXAMPLE"
```

The system returns information like the following:

```
{
  "Targets": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
      "ResourceType": "INSTANCE",
      "Targets": [
        {
          "Key": "InstanceIds",
          "Values": [
            "i-02573cafcfEXAMPLE"
          ]
        }
      ]
    }
  ]
}
```

Continue to Step 3: Register a task with the maintenance window (AWS CLI) (p. 648).
Examples: Register targets with a maintenance window

You can register a single instance as a target using its instance ID, as demonstrated in Step 2: Register a target instance with the maintenance window (AWS CLI) (p. 643). You can also register one or more instances as targets using the command formats on this page.

In general, there are two methods for identifying the instances you want to use as maintenance window targets: specifying individual instances, and using resource tags. The resource tags method provides more options, as shown in examples 2-3.

You can also specify one or more resource groups as the target of a maintenance window. A resource group can include instances and many other types of supported AWS resources. Examples 4 and 5, next, demonstrate how to add resource groups to your maintenance window targets.


For information about limits for the Maintenance Windows capability, in addition to those specified in the following examples, see Systems Manager service quotas in the Amazon Web Services General Reference.

Topics
• Example 1: Register multiple targets using instance IDs (p. 645)
• Example 2: Register targets using resource tags applied to instances (p. 645)
• Example 3: Register targets using a group of tag keys (without tag values) (p. 646)
• Example 4: Register targets using a resource group name (p. 647)

Example 1: Register multiple targets using instance IDs

Run the following command on your local machine format to register multiple instances as targets using their instance IDs:

Linux

```bash
aws ssm register-target-with-maintenance-window \ 
   --window-id "mw-0c50858d01EXAMPLE" \ 
   --resource-type "INSTANCE" \ 
   --target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE"
```

Windows

```bash
aws ssm register-target-with-maintenance-window ^
   --window-id "mw-0c50858d01EXAMPLE ^
   --resource-type "INSTANCE ^
   --target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE"
```

Recommended use: Most useful when registering a unique group of instances with any maintenance window for the first time and they do not share a common instance tag.

Quotas: You can specify up to 50 instances total for each maintenance window target.

Example 2: Register targets using resource tags applied to instances

Run the following command on your local machine to register instances that are all already tagged with a key-value pair you have assigned:

```bash
aws ssm register-target-with-maintenance-window \ 
   --window-id "mw-0c50858d01EXAMPLE" \ 
   --resource-type "INSTANCE" \ 
   --target "Key=tag_key,Values=tag_value"
```
Linux

```bash
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "INSTANCE" \
    --target "Key=tag:Region,Values=East"
```

Windows

```bash
aws ssm register-target-with-maintenance-window ^
    --window-id "mw-0c50858d01EXAMPLE" ^
    --resource-type "INSTANCE" ^
    --target "Key=tag:Region,Values=East"
```

**Recommended use:** Most useful when registering a unique group of instances with any maintenance window for the first time and they do share a common instance tag.

**Quotas:** You can specify up to five key-value pairs total for each target. If you specify more than one key-value pair, an instance must be tagged with all the tag keys and values you specify to be included in the target group.

**Note**
You can tag a group of instances with the tag-key **Patch Group** and assign the instances a common key value, such as `my-patch-group`. Patch Manager evaluates the **Patch Group** key on instances to help determine which patch baseline applies to them. If your task will run the AWS-RunPatchBaseline SSM document (or the legacy AWS-ApplyPatchBaseline SSM document), you can specify the same **Patch Group** key-value pair when you register targets with a maintenance window. For example: `--target "Key=tag:Patch Group,Values=my-patch-group"`. Doing so enables you to easily use a maintenance window to update patches on a group of instances that are already associated with the same patch baseline. For more information, see [About patch groups](p. 953).

**Example 3: Register targets using a group of tag keys (without tag values)**

Run the following command on your local machine to register instances that all have one or more tag keys assigned to them, regardless of their key values.

Linux

```bash
aws ssm register-target-with-maintenance-window \
    --window-id "mw-0c50858d01EXAMPLE" \
    --resource-type "INSTANCE" \
    --target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

Windows

```bash
aws ssm register-target-with-maintenance-window ^
    --window-id "mw-0c50858d01EXAMPLE" ^
    --resource-type "INSTANCE" ^
    --target "Key=tag-key,Values=Name,Instance-Type,CostCenter"
```

**Recommended use:** Useful when you want to target instances by specifying multiple tag keys (without their values) rather than just one tag-key or a tag key-value pair.

**Quotas:** You can specify up to five tag-keys total for each target. If you specify more than one tag key, an instance must be tagged with all the tag keys you specify to be included in the target group.
Example 4: Register targets using a resource group name

Run the following command on your local machine to register a specified resource group, regardless of the type of resources it contains. If the tasks you assign to the maintenance window do not act on a type of resource included in this resource group, the system might report an error. Tasks for which a supported resource type is found continue to run despite these errors.

Linux

```bash
aws ssm register-target-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
   --resource-type "RESOURCE_GROUP"
   --target "Key=resource-groups:Name,Values=MyResourceGroup"
```

Windows

```bash
aws ssm register-target-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
   --resource-type "RESOURCE_GROUP"
   --target "Key=resource-groups:Name,Values=MyResourceGroup"
```

**Recommended use:** Useful when you want to quickly specify a resource group as a target without evaluating whether all of its resource types will be targeted by a maintenance window, or when you know that the resource group contains only the resource types that your tasks perform actions on.

**Quotas:** You can specify only one resource group as a target.

Example 5: Register targets by filtering resource types in a resource group

Run the following command on your local machine to register only certain resource types that belong to a resource group that you specify. With this option, even if you add a task for a resource type that belongs to the resource group, the task won’t run if you haven’t explicitly added the resource type to the filter.

Linux

```bash
aws ssm register-target-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
   --resource-type "RESOURCE_GROUP"
   --target "Key=resource-groups:Name,Values=MyResourceGroup"
   "Key=resource-groups:ResourceTypeFilters,Values=AWS::EC2::Instance,AWS::ECS::Cluster"
```

Windows

```bash
aws ssm register-target-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
   --resource-type "RESOURCE_GROUP"
   --target "Key=resource-groups:Name,Values=MyResourceGroup"
   "Key=resource-groups:ResourceTypeFilters,Values=AWS::EC2::Instance,AWS::ECS::Cluster"
```

**Recommended use:** Useful when you want to maintain strict control over the types of AWS resources your maintenance window can run actions on, or when your resource group contains a large number of resource types and you want to avoid unnecessary error reports in your maintenance window logs.

**Quotas:** You can specify only one resource group as a target.
Step 3: Register a task with the maintenance window (AWS CLI)

In this step of the tutorial, you register a Run Command task that runs the `df` command on your EC2 instance for Linux. The results of this standard Linux command show how much space is free and how much is used on the disk file system of your instance.

-or-

If you are targeting an EC2 instance for Windows Server instead of Linux, replace `df` in the following command with `ipconfig`. Output from this command lists details about the IP address, subnet mask, and default gateway for adapters on the target instance.

When you are ready to register other task types, or use more of the available Run Command options, see Examples: Register tasks with a maintenance window (p. 651). There, we provide more information about all four task types, and some of their most important options, to help you plan for more extensive real-world scenarios.

To register a task with a maintenance window

1. Run the following command on your local machine. The version to run from a local Windows machine includes the escape characters (`) that you need to run the command from your command line tool.

   Linux

   ```bash
   aws ssm register-task-with-maintenance-window
       --window-id mw-0c50858d01EXAMPLE
       --task-arn "AWS-RunShellScript"
       --max-concurrency 1 --max-errors 1
       --priority 10
       --targets "Key=InstanceIds,Values=i-0471e04240EXAMPLE"
       --task-type "RUN_COMMAND"
       --task-invocation-parameters '{"RunCommand":{"Parameters":{"commands":["df"]}}}'
   ```

   Windows

   ```bash
   aws ssm register-task-with-maintenance-window
       --window-id mw-0c50858d01EXAMPLE
       --task-arn "AWS-RunShellScript"
       --max-concurrency 1 --max-errors 1
       --priority 10
       --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE"
       --task-type "RUN_COMMAND"
       --task-invocation-parameters="{"RunCommand":{"Parameters":{"commands":["df"]}}}
   ```

   The system returns information similar to the following:

   ```json
   {
     "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
   }
   ```

2. Now run the following command to view details about the maintenance window task you created.

   Linux

   ```bash
   aws ssm describe-maintenance-window-tasks
   ```
Windows

aws ssm describe-maintenance-window-tasks ^
--window-id mw-0c50858d01EXAMPLE

3. The system returns information similar to the following:

```json
{  
    "Tasks": [  
        {  
            "WindowId": "mw-0c50858d01EXAMPLE",  
            "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",  
            "TaskArn": "AWS-RunShellScript",  
            "Type": "RUN_COMMAND",  
            "Targets": [  
                {  
                    "Key": "InstanceIds",  
                    "Values": [  
                        "i-02573caf9EXAMPLE"
                    ]
                }
            ],  
            "TaskParameters": {},  
            "Priority": 10,  
            "ServiceRoleArn": "arn:aws:iam::123456789012:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",  
            "MaxConcurrency": "1",  
            "MaxErrors": "1"
        }
    ]
}
```

4. Wait until the task has had time to run, based on the schedule you specified in Step 1: Create the maintenance window (AWS CLI) (p. 642). For example, if you specified `--schedule "rate(5 minutes)"`, wait five minutes. Then run the following command to view information about any executions that occurred for this task.

Linux

```bash
aws ssm describe-maintenance-window-executions \  
--window-id mw-0c50858d01EXAMPLE
```

Windows

```bash
aws ssm describe-maintenance-window-executions ^  
--window-id mw-0c50858d01EXAMPLE
```

The system returns information similar to the following:

```json
{  
    "WindowExecutions": [  
        {  
            "WindowId": "mw-0c50858d01EXAMPLE",  
            "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",  
            "Status": "SUCCESS",  
            "StartTime": 1557593493.096,  
            "EndTime": 1557593498.611
        }
    ]
}```
Tip
After the task completes successfully, you can decrease the rate at which the maintenance window runs. For example, run the following command to decrease the frequency to once a week:

Linux

```bash
aws ssm update-maintenance-window \
   --window-id mw-0c50858d01EXAMPLE \
   --schedule "rate(7 days)"
```

Windows

```bash
aws ssm update-maintenance-window ^
   --window-id mw-0c50858d01EXAMPLE ^
   --schedule "rate(7 days)"
```

For information about managing maintenance window schedules, see Reference: Cron and rate expressions for Systems Manager (p. 1189) and Reference: Maintenance window scheduling and active period options (p. 693).

For information about using the AWS CLI to modify a maintenance window, see Tutorial: Update a maintenance window (AWS CLI) (p. 676).

For practice running AWS CLI commands to view more details about your maintenance window task and its executions, continue to Tutorial: View information about tasks and task executions (AWS CLI) (p. 674).

About tutorial command output

It's beyond the scope of this tutorial to use the AWS CLI to view the output of the Run Command command associated with your maintenance window task executions.

You could view this data, however, using the AWS CLI. (You could also view the output in the Systems Manager console or in a log file stored in an S3 bucket, if you had configured the maintenance window to store command output there.) You would find that the output of the `df` command on an EC2 instance for Linux is similar to the following:

```
Filesystem 1K-blocks Used Available Use% Mounted on
devtmpfs 485716 0 485716 0% /dev
tmpfs 503624 0 503624 0% /dev/shm
tmpfs 503624 328 503296 1% /run
tmpfs 503624 0 503624 0% /sys/fs/cgroup
/dev/xvda1 8376300 1464160 6912140 18% /
```

The output of the `ipconfig` command on an EC2 instance for Windows Server is similar to the following:

```
Windows IP Configuration
```

650
Examples: Register tasks with a maintenance window

You can register a Systems Manager Run Command task with a maintenance window using the AWS CLI, as demonstrated in Step 3: Register a task with the maintenance window (AWS CLI) (p. 648). You can also register tasks for Systems Manager Automation workflows, AWS Lambda functions, and AWS Step Functions tasks, as demonstrated below.

In this topic, we provide examples of using the `register-task-with-maintenance-window` CLI command to register each of the four supported task types with a maintenance window. The examples are for demonstration only, but you can modify them to create working task registration commands.

Using the `--cli-input-json` option

To better manage your task options, you can use the command option `--cli-input-json`, with option values referenced in a JSON file.

To use the sample JSON file content we provide in the following examples, do the following on your local machine:

1. Create a file with a name such as `MyRunCommandTask.json`, `MyAutomationTask.json`, or another name that you prefer.
2. Copy the contents of our JSON sample into the file.
3. Modify the contents of the file for your task registration, and then save the file.
4. In the same directory where you stored the file, run the following command. Substitute your file name for `MyFile.json`.

   ![aws ssm register-task-with-maintenance-window --cli-input-json: MyFile.json](attachment:image)
Windows

```bash
aws ssm register-task-with-maintenance-window ^
   --cli-input-json file://MyFile.json
```

About pseudo parameters

In some examples, we use pseudo parameters as the method to pass ID information to your tasks. For instance, `{RESOURCE_ID}` is used to pass IDs of AWS resources to Automation, Lambda, and Step Functions tasks. For more information about pseudo parameters in `--task-invocation-parameters` content, see About pseudo parameters (p. 659).

More information

For information about some fundamental `register-task-with-maintenance-window` options, see About register-task-with-maintenance-windows options (p. 656).

For comprehensive information about command options, see the following topics:

- `register-task-with-maintenance-window` in the AWS CLI Command Reference
- RegisterTaskWithMaintenanceWindow in the AWS Systems Manager API Reference

Task registration examples

The following sections provide a sample AWS CLI command for registering a supported task type and a JSON sample that can be used with the `--cli-input-json` option.

Register a Systems Manager Run Command task

The following examples demonstrate how to register Systems Manager Run Command tasks with a maintenance window using the AWS CLI.

**AWS CLI command:**

**Linux**

```bash
aws ssm register-task-with-maintenance-window \
   --window-id mw-0c50858d01EXAMPLE \
   --task-arn "AWS-RunShellScript" \
   --max-concurrency 1 --max-errors 1 --priority 10 \
   --targets "Key=InstanceIds,Values=i-02573caf6EXAMPLE" \
   --task-type "RUN_COMMAND" \
   --task-invocation-parameters "{"RunCommand":{"Parameters":{"commands":["df"]}}}" \
```

**Windows**

```bash
aws ssm register-task-with-maintenance-window ^
   --window-id mw-0c50858d01EXAMPLE ^
   --task-arn "AWS-RunShellScript" ^
   --max-concurrency 1 --max-errors 1 --priority 10 ^
   --targets "Key=InstanceIds,Values=i-02573caf6EXAMPLE" ^
   --task-type "RUN_COMMAND" ^
   --task-invocation-parameters "{"RunCommand":{"Parameters":{"commands":["df"]}}}" 
```

**JSON content to use with `--cli-input-json` file option:**

```json
"{"RunCommand":{"Parameters":{"commands":["df"]}}}
```
AWS Systems Manager User Guide
Maintenance Windows tutorials (AWS CLI)

Register a Systems Manager Automation task

The following examples demonstrate how to register Systems Manager Automation tasks with a maintenance window using the AWS CLI:

**AWS CLI command:**

**Linux**

```bash
aws ssm register-task-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
   --targets Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE
   --task-arn "AWS-RestartEC2Instance"
   --service-role-arn arn:aws:iam::123456789012:role/MyMaintenanceWindowServiceRole
   --task-type AUTOMATION
   --task-invocation-parameters
"Automation={DocumentVersion=5,Parameters={instanceId='{{RESOURCE_ID}}'}}"
   --priority 0 --max-concurrency 10 --max-errors 5 --name "My-Restart-EC2-Instances-Automation-Task"
   --description "Automation task to restart EC2 instances"
```

**Windows**

```bash
aws ssm register-task-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
```

---

Register a Systems Manager Automation task

The following examples demonstrate how to register Systems Manager Automation tasks with a maintenance window using the AWS CLI:

**AWS CLI command:**

**Linux**

```bash
aws ssm register-task-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
   --targets Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE
   --task-arn "AWS-RestartEC2Instance"
   --service-role-arn arn:aws:iam::123456789012:role/MyMaintenanceWindowServiceRole
   --task-type AUTOMATION
   --task-invocation-parameters
"Automation={DocumentVersion=5,Parameters={instanceId='{{RESOURCE_ID}}'}}"
   --priority 0 --max-concurrency 10 --max-errors 5 --name "My-Restart-EC2-Instances-Automation-Task"
   --description "Automation task to restart EC2 instances"
```

**Windows**

```bash
aws ssm register-task-with-maintenance-window
   --window-id "mw-0c50858d01EXAMPLE"
```
Maintenance Windows tutorials (AWS CLI)

JSON content to use with `--cli-input-json` file option:

```json
{
  "WindowId": "mw-0c50858d01EXAMPLE",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
      ]
    }
  ],
  "TaskArn": "AWS-PatchInstanceWithRollback",
  "TaskType": "AUTOMATION",
  "MaxConcurrency": "10",
  "MaxErrors": "10",
  "TaskInvocationParameters": {
    "Automation": {
      "DocumentVersion": "1",
      "Parameters": {
        "instanceId": [
          "{{RESOURCE_ID}}"
        ]
      }
    }
  }
}
```

Register an AWS Lambda task

The following examples demonstrate how to register AWS Lambda function tasks with a maintenance window using the AWS CLI.

For these examples, the user who created the Lambda function named it `SSMrestart-my-instances` and created two parameters called `instanceId` and `targetType`.

**Important**

The IAM policy for Maintenance Windows requires that you prefix Lambda function (or alias) names with SSM. Before you proceed to register this type of task, you must update its name in AWS Lambda to include SSM. For example, if your Lambda function name is `MyLambdaFunction`, change it to `SSMMyLambdaFunction`.

**AWS CLI command:**

Linux

```bash
aws ssm register-task-with-maintenance-window \
  --window-id "mw-0c50858d01EXAMPLE" \
  --targets "Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE" \
  --priority 2 --max-concurrency 10 --max-errors 5 --name "My-Lambda-Example" \
  --description "A description for my LAMBDA example task" --task-type "LAMBDA" \
```
Windows

```bash
aws ssm register-task-with-maintenance-window
  --window-id "mw-0c50858d01EXAMPLE"
  --targets "Key=WindowTargetIds,Values=e32eebcb2-646c-4f4b-8ed1-205fbEXAMPLE" 
  --priority 2 --max-concurrency 10 --max-errors 5 --name "My-Lambda-Example" 
  --description "A description for my LAMBDA example task" --task-type "LAMBDA" 
  --task-invocation-parameters '{"Lambda":{"Payload":{"instanceId": "{{RESOURCE_ID}}","targetType": "{{TARGET_TYPE}}"},"Qualifier": "$LATEST"}}'
```

JSON content to use with `--cli-input-json` file option:

```json
{
    "WindowId": "mw-0c50858d01EXAMPLE",
    "Targets": [
        {
            "Key": "WindowTargetIds",
            "Values": [
                "e32eebcb2-646c-4f4b-8ed1-205fbEXAMPLE"
            ]
        }
    ],
    "TaskArn": "SSM_RestartMyInstances",
    "TaskType": "LAMBDA",
    "MaxConcurrency": "10",
    "MaxErrors": "10",
    "TaskInvocationParameters": {
        "Lambda": {
            "ClientContext": "ew0KICAi--truncated--0KIEXAMPLE",
            "Payload": "{"\"instanceId\": "{{RESOURCE_ID}}", \"targetType\": "{{TARGET_TYPE}}" },\"Qualifier": "$LATEST""
        }
    },
    "Name": "My-Lambda-Task",
    "Description": "A description for my LAMBDA task",
    "Priority": 5
}
```

Register an AWS Step Functions task

The following examples demonstrate how to register AWS Step Functions state machine tasks with a maintenance window using the AWS CLI.

For these examples, the user who created the Step Functions state machine created a state machine named `SSMMyStateMachine` with a parameter called `instanceId`.

**Important**

The IAM policy for Maintenance Windows requires that you prefix Step Functions state machine names with SSM. Before you proceed to register this type of task, you must update its name in AWS Step Functions to include SSM. For example, if your state machine name is `MyStateMachine`, change it to `SSMMyStateMachine`.

**AWS CLI command:**
Linux

```
aws ssm register-task-with-maintenance-window \
  --window-id "mw-0c50858d01EXAMPLE" \
  --targets "Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE" \
  --task-arn arn:aws:states:us-east-2:123456789012:stateMachine:SSMMyStateMachine-
  MggigEXAMPLE \
  --task-type STEP_FUNCTIONS \
  --task-invocation-parameters '{"StepFunctions":{"Input":"\"instanceId\":\"{{RESOURCE_ID}}\""}, "Name": "{INVOCATION_ID}"}' \
  --priority 0 --max-concurrency 10 --max-errors 5 \
  --name "My-Step-Functions-Task" --description "A description for my Step Functions task"
```

Windows

```
aws ssm register-task-with-maintenance-window ^
  --window-id "mw-0c50858d01EXAMPLE" ^
  --targets "Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE" ^
  --task-arn arn:aws:states:us-east-2:123456789012:stateMachine:SSMMyStateMachine-
  MggigEXAMPLE \\^ 
  --task-type STEP_FUNCTIONS ^
  --task-invocation-parameters '{"StepFunctions":{"Input":"\"instanceId\":\"{{RESOURCE_ID}}\""}, "Name": "{INVOCATION_ID}"}' ^
  --priority 0 --max-concurrency 10 --max-errors 5 ^
  --name "My-Step-Functions-Task" --description "A description for my Step Functions task"
```

### JSON content to use with `--cli-input-json` file option:

```json
{
  "WindowId": "mw-0c50858d01EXAMPLE",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
      ]
    }
  ],
  "TaskArn": "SSM_MyStateMachine",
  "TaskType": "STEP_FUNCTIONS",
  "MaxConcurrency": "10",
  "MaxErrors": "10",
  "TaskInvocationParameters": {
    "StepFunctions": {
      "Input": "\"instanceId\":\"{{TARGET_ID}}\"",
      "Name": "{INVOCATION_ID}""
    }
  },
  "Name": "My-Step-Functions-Task",
  "Description": "A description for my Step Functions task",
  "Priority": 5
}
```

### About register-task-with-maintenance-windows options

The `register-task-with-maintenance-window` command provides several options for configuring a task according to your needs. Some are required, some are optional, and some apply to only a single maintenance window task type.
This topic provides information about some of these options to help you work with samples in this tutorial section. For information about all command options, see `register-task-with-maintenance-window` in the **AWS CLI Command Reference**.

**About the `--task-arn` option**

The option `--task-arn` is used to specify the resource that the task uses during execution. The value that you specify depends on the type of task you are registering, as described in the following table.

**TaskArn formats for maintenance window tasks**

<table>
<thead>
<tr>
<th>Maintenance window task type</th>
<th>TaskArn value</th>
</tr>
</thead>
</table>
| **RUN_COMMAND** and **AUTOMATION** | TaskArn is the SSM document name or ARN. For example:  
  AWS-RunBatchShellScript  
  -or-  
| **LAMBDA** | TaskArn is the function name or ARN. For example:  
  SSMMy-Lambda-Function  
  -or-  
| **STEP_FUNCTIONS** | TaskArn is the state machine ARN. For example:  
  -or-  

**Important**

The IAM policy for maintenance windows requires that you prefix Lambda function (or alias) names with `SSM`. Before you register this type of task, you must update its name in AWS Lambda to include `SSM`. For example, if your Lambda function name is `MyLambdaFunction`, change it to `SSMMyLambdaFunction`.

**Important**

The IAM policy for maintenance windows requires that you prefix Step Functions state machine names with `SSM`. Before you register this type of task, you must update its name in AWS Step Functions to include `SSM`. For example, if your state machine name is `MyStateMachine`, change it to `SSMMyStateMachine`.

**About the `--service-role-arn` option**

657
The role for Systems Manager to assume when running the maintenance window task.

Specifying a service role ARN is optional. If you do not specify a service role ARN, Systems Manager creates a service-linked role or uses your account's service-linked role.

Note that the service-linked role for Systems Manager doesn't provide the permissions needed for all scenarios. For more information, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622)

### About the `--task-invocation-parameters` option

The `--task-invocation-parameters` option is used to specify the parameters that are unique to each of the four task types. The supported parameters for each of the four task types are described in the following table.

**Note**

For information about using pseudo parameters in `--task-invocation-parameters` content, such as `{{TARGET_ID}}`, see About pseudo parameters (p. 659).

### Task invocation parameters options for maintenance window tasks

<table>
<thead>
<tr>
<th>Maintenance window task type</th>
<th>Available parameters</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN_COMMAND</td>
<td>Comment</td>
<td>&quot;TaskInvocationParameters&quot;: {</td>
</tr>
<tr>
<td></td>
<td>DocumentHash</td>
<td>&quot;RunCommand&quot;: {</td>
</tr>
<tr>
<td></td>
<td>DocumentHashType</td>
<td>&quot;Comment&quot;: &quot;My</td>
</tr>
<tr>
<td></td>
<td>NotificationConfig</td>
<td>Run Command task comment&quot;,</td>
</tr>
<tr>
<td></td>
<td>OutputS3BucketName</td>
<td>&quot;DocumentHash&quot;:</td>
</tr>
<tr>
<td></td>
<td>OutPutS3KeyPrefix</td>
<td>&quot;6554ed3d--truncated--5EXAMPLE&quot;,</td>
</tr>
<tr>
<td></td>
<td>Parameters</td>
<td>&quot;DocumentHashType&quot;:</td>
</tr>
<tr>
<td></td>
<td>ServiceRoleArn</td>
<td>&quot;Sha256&quot;,</td>
</tr>
<tr>
<td></td>
<td>TimeoutSeconds</td>
<td>&quot;NotificationConfig&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;NotificationArn&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;NotificationEvents&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;FAILURE&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;NotificationType&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Invocation&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;OutputS3BucketName&quot;: &quot;my-s3-bucket-name&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;OutputS3KeyPrefix&quot;: &quot;my-s3-bucket-folder-name&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Parameters&quot;: {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;commands&quot;: [</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;Get-ChildItem$env: temp-Recurse</td>
</tr>
<tr>
<td></td>
<td></td>
<td>]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>
### Maintenance window task type | Available parameters | Example
---|---|---

**AUTOMATION** | DocumentVersion | "TaskInvocationParameters": 
Parameters | 
| | | 
| | | "Automation": { 
| | | "DocumentVersion": "3", 
| | | "Parameters": { 
| | | "instanceid": [ 
| | | "{{TARGET_ID}}" ] 
| | | } 
| | | } 

**LAMBDA** | ClientContext | "TaskInvocationParameters": 
Payload | 
| | | 
| | | "Lambda": { 
| | | "ClientContext": "ew0KICAi-- truncated--0XIEXAMPLE", 
| | | "Payload": "\"\"targetId\": \"{{TARGET_ID}}\", 
| | | "targetType\": \"{{TARGET_TYPE}}\" \"", 
| | | "Qualifier": "$LATEST" 
| | | } 

**STEP_FUNCTIONS** | Input | "TaskInvocationParameters": 
Name | 
| | | 
| | | "StepFunctions": { 
| | | "Input": "\"targetId\": \"{{TARGET_ID}}\", 
| | | "Name": "{{INVOCATION_ID}}" 
| | | } 

---

**About pseudo parameters**

When you register a task, you use the --task-invocation-parameters option to specify the parameters that are unique to each of the four task types. You can also reference certain values using pseudo parameter syntax, such as {{RESOURCE_ID}}, {{TARGET_TYPE}}, and {{WINDOW_TARGET_ID}}. When the maintenance window task runs, it passes the correct values instead
of the pseudo parameter placeholders. The full list of pseudo parameters you can use is provided in Supported pseudo parameters (p. 661).

**Important**
For the target type RESOURCE_GROUP, depending on the ID format needed for the task, you can choose between using {{TARGET_ID}} and {{RESOURCE_ID}} to reference the resource when your task runs. {{TARGET_ID}} returns the full ARN of the resource. {{RESOURCE_ID}} returns only a shorter name or ID of the resource, as shown in these examples.

- {{TARGET_ID}} format: arn:aws:ec2:us-east-1:123456789012:instance/i-02573cafcfEXAMPLE
- {{RESOURCE_ID}} format: i-02573cafcfEXAMPLE

For target type INSTANCE, both the {{TARGET_ID}} and {{RESOURCE_ID}} parameters yield the instance ID only. For more information, see Supported pseudo parameters (p. 661).

**Pseudo parameter examples**
Suppose that your payload for a Lambda task needs to reference an instance by its ID.

Whether you’re using an INSTANCE or a RESOURCE_GROUP maintenance window target, this can be achieved by using the {{RESOURCE_ID}} pseudo parameter:

```json
"TaskType": "LAMBDA",
"TaskInvocationParameters": {
  "Lambda": {
    "ClientContext": "ew0KICAi--truncated--0KIEXAMPLE",
    "Payload": "{"\"instanceId\": \"{{RESOURCE_ID}}\" }",
    "Qualifier": "$LATEST"
  }
}
```

If your Lambda task is intended to run against another supported target type in addition to EC2 instances, such as a DynamoDB table, the same syntax can be used, and {{RESOURCE_ID}} yields the name of the table only. However, if you require the full ARN of the table, use {{TARGET_ID}}, as shown in the following example.

```json
"TaskType": "LAMBDA",
"TaskInvocationParameters": {
  "Lambda": {
    "ClientContext": "ew0KICAi--truncated--0KIEXAMPLE",
    "Payload": "{"\"tableArn\": \"{{TARGET_ID}}\" }",
    "Qualifier": "$LATEST"
  }
}
```

The same syntax works for targeting instances or other resource types. When multiple resource types have been added to a resource group, the task runs against each of the appropriate resources.

**Important**
Not all resource types that might be included in a resource group yield a value for the {{RESOURCE_ID}} parameter. For a list of supported resource types, see Supported pseudo parameters (p. 661).

As another example, to run an Automation task that stops your EC2 instances, you specify the AWS-StopEC2Instance SSM document as the TaskArn value and use the {{RESOURCE_ID}} pseudo parameter:
To run an Automation task that copies a snapshot of an Amazon Elastic Block Store (Amazon EBS) volume, you specify the `AWS-CopySnapshot` SSM document as the TaskArn value and use the `{{RESOURCE_ID}}` pseudo parameter:

```
"TaskArn": "AWS-CopySnapshot",
"TaskType": "AUTOMATION"
"TaskInvocationParameters": {
  "Automation": {
    "DocumentVersion": "1",
    "Parameters": {
      "SourceRegion": "us-east-2",
      "targetType": "RESOURCE_GROUP",
      "SnapshotId": ["{{RESOURCE_ID}}"]
    }
  }
}
```

Supported pseudo parameters

The following list describes the pseudo parameters that you can specify using the `{{PSEUDO_PARAMETER}}` syntax in the `--task-invocation-parameters` option.

- **WINDOW_ID**: The ID of the target maintenance window.
- **WINDOW_TASK_ID**: The ID of the window task that is executing.
- **WINDOW_TARGET_ID**: The ID of the window target that includes the target (target ID).
- **WINDOW_EXECUTION_ID**: The ID of the current window execution.
- **TASK_EXECUTION_ID**: The ID of the current task execution.
- **INVOCATION_ID**: The ID of the current invocation.
- **TARGET_TYPE**: The type of target. Supported types include `RESOURCE_GROUP` and `INSTANCE`.
- **TARGET_ID**:

  If the target type you specify is `INSTANCE`, the TARGET_ID pseudo parameter is replaced by the ID of the instance; for example, `i-078a280217EXAMPLE`.

  If the target type you specify is `RESOURCE_GROUP`, the value referenced for the task execution is the full ARN of the resource; for example: `arn:aws:ec2:us-east-1:123456789012:instance/i-078a280217EXAMPLE`. The following table provides sample TARGET_ID values for particular resource types in a resource group.
<table>
<thead>
<tr>
<th>Resource type</th>
<th>Example RESOURCE_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS::EC2::Instance</td>
<td>i-078a280217EXAMPLE</td>
</tr>
<tr>
<td>AWS::EC2::Image</td>
<td>ami-02250b3732EXAMPLE</td>
</tr>
<tr>
<td>AWS::EC2::SecurityGroup</td>
<td>sg-cEXAMPLE</td>
</tr>
<tr>
<td>AWS::EC2::Snapshot</td>
<td>snap-03866bf003EXAMPLE</td>
</tr>
<tr>
<td>AWS::EC2::Volume</td>
<td>vol-0912e04d78EXAMPLE</td>
</tr>
<tr>
<td>AWS::DynamoDB::Table</td>
<td>MyTable</td>
</tr>
<tr>
<td>AWS::S3::Bucket</td>
<td>MyBucket</td>
</tr>
</tbody>
</table>

**Note**

If the AWS resource group you specify includes resource types that do not yield a RESOURCE_ID value, and are not listed in the table above, then the RESOURCE_ID parameter is not populated. An execution invocation will still occur for that resource. In these cases, use the TARGET_ID pseudo parameter instead, which will be replaced with the full ARN of the resource.
Tutorial: View information about maintenance windows (AWS CLI)

This tutorial includes commands to help you update or get information about your maintenance windows, tasks, executions, and invocations. The examples are organized by command to demonstrate how to use command options to filter for the type of detail you want to see.

As you follow the steps in this tutorial, replace the values in italicized red text with your own options and IDs. For example, replace the maintenance window ID mw-0c50858d01EXAMPLE and the instance ID i-02573cafcfEXAMPLE with IDs of resources you create.

For information about setting up and configuring the CLI, see Installing the AWS Command Line Interface and Configuring the AWS Command Line Interface.

Command Examples

- Examples for 'describe-maintenance-windows' (p. 663)
- Examples for 'describe-maintenance-window-targets' (p. 665)
- Examples for 'describe-maintenance-window-tasks' (p. 666)
- Examples for 'describe-maintenance-windows-for-target' (p. 669)
- Examples for 'describe-maintenance-window-executions' (p. 669)
- Examples for 'describe-maintenance-window-schedule' (p. 671)

Examples for 'describe-maintenance-windows'

List all maintenance windows in your AWS account

Run the following command:

```
aws ssm describe-maintenance-windows
```

The system returns information like the following:

```json
{
   "WindowIdentities": [
      {
         "WindowId": "mw-0c50858d01EXAMPLE",
         "Name": "My-First-Maintenance-Window",
         "Enabled": true,
         "Duration": 2,
         "Cutoff": 0,
         "NextExecutionTime": "2019-05-18T17:01:01.137Z"
      },
      {
         "WindowId": "mw-9a8b7c6d5eEXAMPLE",
         "Name": "My-Second-Maintenance-Window",
         "Enabled": true,
         "Duration": 4,
         "Cutoff": 1,
         "NextExecutionTime": "2019-05-30T03:30:00.137Z"
      }
   ]
}
```

List all enabled maintenance windows
Run the following command:

```bash
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=true"
```

The system returns information like the following:

```json
{
    "WindowIdentities": [
        {
            "WindowId": "mw-0c508580d01EXAMPLE",
            "Name": "My-First-Maintenance-Window",
            "Enabled": true,
            "Duration": 2,
            "Cutoff": 0,
            "NextExecutionTime": "2019-05-18T17:01:01.137Z"
        },
        {
            "WindowId": "mw-9a8b7c6d5eEXAMPLE",
            "Name": "My-Second-Maintenance-Window",
            "Enabled": true,
            "Duration": 4,
            "Cutoff": 1,
            "NextExecutionTime": "2019-05-30T03:30:00.137Z"
        }
    ]
}
```

**List all disabled maintenance windows**

Run the following command:

```bash
aws ssm describe-maintenance-windows --filters "Key=Enabled,Values=false"
```

The system returns information like the following:

```json
{
    "WindowIdentities": [
        {
            "WindowId": "mw-6e5c9d4b7cEXAMPLE",
            "Name": "My-Disabled-Maintenance-Window",
            "Enabled": false,
            "Duration": 2,
            "Cutoff": 1
        }
    ]
}
```

**List all maintenance windows having names that start with a certain prefix**

Run the following command:

```bash
aws ssm describe-maintenance-windows --filters "Key=Name,Values=My"
```

The system returns information like the following:

```json
{
    "WindowIdentities": [
```
Examples for 'describe-maintenance-window-targets'

Display the targets for a maintenance window matching a specific owner information value

Run the following command:

**Linux**

```
aws ssm describe-maintenance-window-targets \
  --window-id "mw-6e5c9d4b7cEXAMPLE" \
  --filters "Key=OwnerInformation,Values=CostCenter1"
```

**Windows**

```
aws ssm describe-maintenance-window-targets ^ \
  --window-id "mw-6e5c9d4b7cEXAMPLE" ^ \
  --filters "Key=OwnerInformation,Values=CostCenter1"
```

**Note**
The supported filter keys are Type, WindowTargetId and OwnerInformation.

The system returns information like the following:

```
{
  "Targets": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE",
      "ResourceType": "INSTANCE",
      "Targets": [
        {
          "Key": "tag:Name",
          "Values": [
        ```
Examples for 'describe-maintenance-window-tasks'

Show all registered tasks that invoke the AWS-RunPowerShellScript Run Command

Run the following command:

Linux

```bash
aws ssm describe-maintenance-window-tasks \
--window-id "mw-0c5085d01EXAMPLE" \
--filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

Windows

```bash
aws ssm describe-maintenance-window-tasks ^
--window-id "mw-0c5085d01EXAMPLE" ^
--filters "Key=TaskArn,Values=AWS-RunPowerShellScript"
```

The system returns information like the following:

```json
{
  "Tasks": [  
    {
      "ServiceRoleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "MaxErrors": "1",
      "TaskArn": "AWS-RunPowerShellScript",
      "MaxConcurrency": "1",
      "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
      "TaskParameters": {
        "commands": {
          "Values": ["driverquery.exe"]
        }
      },
      "Priority": 3,
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "TaskTargetId": "i-02573caf01EXAMPLE",
          "TaskTargetType": "INSTANCE"
        }
      ]
    },
    {
      "ServiceRoleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "MaxErrors": "1",
      "TaskArn": "AWS-RunPowerShellScript",
      "MaxConcurrency": "1",
      "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
      "TaskParameters": {
        "commands": {
          "Values": ["driverquery.exe"]
        }
      },
      "Priority": 3,
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "TaskTargetId": "i-02573caf01EXAMPLE",
          "TaskTargetType": "INSTANCE"
        }
      ]
    }
  ]
}
```
Show all registered tasks that have a priority of "3"

Run the following command:

Linux

```bash
aws ssm describe-maintenance-window-tasks
--window-id "mw-9a8b7c6d5eEXAMPLE"
--filters "Key=Priority,Values=3"
```

Windows

```bash
aws ssm describe-maintenance-window-tasks
--window-id "mw-9a8b7c6d5eEXAMPLE"
--filters "Key=Priority,Values=3"
```

The system returns information like the following:

```json
{
  "Tasks": [ 
    
    
    ]
}
```
Show all registered tasks that have a priority of "1" and use Run Command

Run the following command:

**Linux**

```bash
aws ssm describe-maintenance-window-tasks \
  --window-id "mw-0c50858d01EXAMPLE" \
  --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

**Windows**

```bash
aws ssm describe-maintenance-window-tasks ^
  --window-id "mw-0c50858d01EXAMPLE" ^
  --filters "Key=Priority,Values=1" "Key=TaskType,Values=RUN_COMMAND"
```

The system returns information like the following:

```
{
  "Tasks": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
      "TaskArn": "AWS-RunShellScript",
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "Key": "InstanceIds",
          "Values": [
            "i-02573cafcfEXAMPLE"
          ]
        }
      ],
      "TaskParameters": {},
      "Priority": 1,
      "ServiceRoleArn": "arn:aws:iam::11112223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "MaxConcurrency": "1",
      "MaxErrors": "1"
    },
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowTaskId": "8a5c4629-31b0-4edd-8aea-33698EXAMPLE",
      "TaskArn": "AWS-UpdateSSMAgent",
      "Type": "RUN_COMMAND",
      "Targets": [
        {
          "Key": "InstanceIds",
          "Values": [
            "i-0471e04240EXAMPLE"
          ]
        }
      ],
      "TaskParameters": {},
      "Priority": 1,
      "ServiceRoleArn": "arn:aws:iam::11112223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
      "MaxConcurrency": "1",
      "MaxErrors": "1"
    }
  ]
}
```
"ServiceRoleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
"MaxConcurrency": "1",
"MaxErrors": "1",
"Name": "My-Run-Command-Task",
"Description": "My Run Command task to update SSM Agent on an instance"
]}
]

Examples for 'describe-maintenance-windows-for-target'

List information about the maintenance window targets or tasks associated with a specific instance

Run the following command:

Linux

```bash
aws ssm describe-maintenance-windows-for-target \
--resource-type INSTANCE \
--targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE" \
--max-results 10
```

Windows

```bash
aws ssm describe-maintenance-windows-for-target ^ 
--resource-type INSTANCE ^ 
--targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE" ^ 
--max-results 10
```

The system returns information like the following:

```json
{
    "WindowIdentities": [
        {
            "WindowId": "mw-0c50858d01EXAMPLE",
            "Name": "My-First-Maintenance-Window"
        },
        {
            "WindowId": "mw-9a8b7c6d5eEXAMPLE",
            "Name": "My-Second-Maintenance-Window"
        }
    ]
}
```

Examples for 'describe-maintenance-window-executions'

List all tasks run before a certain date

Run the following command:

Linux

```bash
aws ssm describe-maintenance-window-executions \
--window-id "mw-9a8b7c6d5eEXAMPLE" \
--filters "Key=ExecutedBefore,Values=2019-05-12T05:00:00Z"
```
Windows

aws ssm describe-maintenance-window-executions ^
--window-id "mw-9a8b7c6d5eEXAMPLE" ^
--filters "Key=ExecutedBefore,Values=2019-05-12T05:00:00Z"

The system returns information like the following:

```
{
  "WindowExecutions": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
      "Status": "FAILED",
      "StatusDetails": "The following SSM parameters are invalid: LevelUp",
      "StartTime": 1557617747.993,
      "EndTime": 1557617748.101
    },
    {
      "WindowId": "mw-9a8b7c6d5eEXAMPLE",
      "WindowExecutionId": "791b72e0-f0da-4021-8b35-f95dfEXAMPLE",
      "Status": "SUCCESS",
      "StartTime": 1557594085.428,
      "EndTime": 1557594090.978
    },
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowExecutionId": "ecec60fa-6bb0-4d26-98c7-140308EXAMPLE",
      "Status": "SUCCESS",
      "StartTime": 1557593793.483,
      "EndTime": 1557593798.978
    }
  ]
}
```

List all tasks run after a certain date

Run the following command:

Linux

aws ssm describe-maintenance-window-executions \
--window-id "mw-9a8b7c6d5eEXAMPLE" \
--filters "Key=ExecutedAfter,Values=2018-12-31T17:00:00Z"

Windows

aws ssm describe-maintenance-window-executions ^
--window-id "mw-9a8b7c6d5eEXAMPLE" ^
--filters "Key=ExecutedAfter,Values=2018-12-31T17:00:00Z"

The system returns information like the following:

```
{
  "WindowExecutions": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
```
Examples for 'describe-maintenance-window-schedule'

Display the next ten scheduled maintenance window runs for a particular instance

Run the following command:

Linux

```bash
aws ssm describe-maintenance-window-schedule \
  --resource-type INSTANCE \
  --targets "Key=InstanceIds,Values=i-07782c72faEXAMPLE" \
  --max-results 10
```

Windows

```bash
aws ssm describe-maintenance-window-schedule ^
  --resource-type INSTANCE ^
  --targets "Key=InstanceIds,Values=i-07782c72faEXAMPLE" ^
  --max-results 10
```

The system returns information like the following:

```json
{
  "ScheduledWindowExecutions": [
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "Name": "My-First-Maintenance-Window",
      "ExecutionTime": "2019-05-18T23:35:24.902Z"
    },
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "Name": "My-First-Maintenance-Window",
      "ExecutionTime": "2019-05-25T23:35:24.902Z"
    },
    {
      "WindowId": "mw-0c50858d01EXAMPLE",
      "Name": "My-First-Maintenance-Window",
      "ExecutionTime": "2019-06-01T23:35:24.902Z"
  ]
}
```
Display the maintenance window schedule for instances tagged with a certain key-value pair

Run the following command:

**Linux**

```bash
aws ssm describe-maintenance-window-schedule
  --resource-type INSTANCE
  --targets "Key=tag:prod,Values=rhel7"
```

**Windows**

```bash
aws ssm describe-maintenance-window-schedule
  --resource-type INSTANCE
  --targets "Key=tag:prod,Values=rhel7"
```

The system returns information like the following:

```json
{
  "ScheduledWindowExecutions": [
  
  ]
}
```
Display start times for next four runs of a maintenance window

Run the following command:

**Linux**

```bash
aws ssm describe-maintenance-window-schedule \
--window-id "mw-0c50858d01EXAMPLE" \
--max-results "4"
```

**Windows**

```bash
aws ssm describe-maintenance-window-schedule ^
--window-id "mw-0c50858d01EXAMPLE" ^
--max-results "4"
```

The system returns information like the following:

```json
{
    "WindowSchedule": [
        {
            "ScheduledWindowExecutions": [
                {
                    "ExecutionTime": "2019-10-04T10:10:10Z",
                    "Name": "My-First-Maintenance-Window",
                    "WindowId": "mw-0c50858d01EXAMPLE"
                },
                {
                    "ExecutionTime": "2019-10-11T10:10:10Z",
                    "Name": "My-First-Maintenance-Window",
                    "WindowId": "mw-0c50858d01EXAMPLE"
                }
            ]
        }
    ]
}
```
Tutorial: View information about tasks and task executions (AWS CLI)

This tutorial demonstrates how to use the AWS CLI to view details about your completed maintenance window task executions.

If you are continuing directly from Tutorial: Create and configure a maintenance window (AWS CLI) (p. 642), make sure you have allowed enough time for your maintenance window to run at least once in order to see its execution results.

As you follow the steps in this tutorial, replace the values in italicized red text with your own options and IDs. For example, replace the maintenance window ID mw-0c50858d01EXAMPLE and the instance ID i-02573cafcfEXAMPLE with IDs of resources you create.

To view information about tasks and task executions (AWS CLI)

1. Run the following command to view a list of task executions for a specific maintenance window:

   ```bash
   aws ssm describe-maintenance-window-executions --window-id "mw-0c50858d01EXAMPLE"
   ```

   The system returns information like the following:

   ```json
   { 
     "WindowExecutions": [ 
       { 
         "WindowId": "mw-0c50858d01EXAMPLE", 
         "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE", 
         "Status": "SUCCESS", 
         "StartTime": 1557593793.483, 
         "EndTime": 1557593798.978 
       }, 
       { 
         "WindowId": "mw-0c50858d01EXAMPLE", 
         "WindowExecutionId": "791b72e0-f0da-4021-8b35-f95dfEXAMPLE", 
         "Status": "SUCCESS", 
         "StartTime": 1557593493.096, 
         "EndTime": 1557593498.611 
       }, 
       { 
         "WindowId": "mw-0c50858d01EXAMPLE", 
         "WindowExecutionId": "ecec60fa-6bb0-4d26-98c7-140308EXAMPLE", 
         "Status": "SUCCESS", 
         "StatusDetails": "No tasks to execute.", 
         "StartTime": 1557593193.309, 
         "EndTime": 1557593193.334 
       } 
   ] 
   ```
2. Run the following command to get information about a maintenance window task execution:

```bash
aws ssm get-maintenance-window-execution \
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE"
```

The system returns information like the following:

```json
{
"WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
"TaskIds": [
"c9b05aba-197f-4d8d-be34-e73fbEXAMPLE"
],
"Status": "SUCCESS",
"StartTime": 1557593493.096,
"EndTime": 1557593498.611
}
```

3. Run the following command to list the tasks run as part of a maintenance window execution:

```bash
aws ssm describe-maintenance-window-execution-tasks \
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE"
```

The system returns information like the following:

```json
{
"WindowExecutionTaskIdentities": [
{
"WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
"TaskExecutionId": "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE",
"Status": "SUCCESS",
"StartTime": 1557593493.162,
"EndTime": 1557593498.57,
"TaskArn": "AWS-RunShellScript",
"TaskType": "RUN_COMMAND"
}
]
```

4. Run the following command to get the details of a task execution:

```bash
aws ssm get-maintenance-window-execution-task \
--window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE" \
--task-id "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE"
```

The system returns information like the following:

```json
{
"WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
"TaskExecutionId": "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE",
"TaskArn": "AWS-RunShellScript",
"ServiceRole": "arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/\nAWSServiceRoleForAmazonSSM",
"Type": "RUN_COMMAND",
"TaskParameters": {
"aws:InstanceId": 
```
5. Run the following command to get the specific task invocations performed for a task execution.

**Linux**
```
aws ssm describe-maintenance-window-execution-task-invocations \
  --window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE" \
  --task-id "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE"
```

**Windows**
```
aws ssm describe-maintenance-window-execution-task-invocations ^ \
  --window-execution-id "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE" ^ \
  --task-id "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE"
```

The system returns information like the following.
```
{
  "WindowExecutionTaskInvocationIdentities": [
    {
      "WindowExecutionId": "14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE",
      "TaskExecutionId": "c9b05aba-197f-4d8d-be34-e73fbEXAMPLE",
      "InvocationId": "c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
      "ExecutionId": "76a5a04f-caf6-490c-b448-92c02EXAMPLE",
      "TaskType": "RUN_COMMAND",
      "Parameters": "{"documentName":"AWS-RunShellScript","instanceIds":\n["i-02573cafcfEXAMPLE"],\n"maxConcurrency":\n"1","maxErrors":\n"1","parameters":\n{"commands":\n["df"]})",
      "Status": "SUCCESS",
      "StatusDetails": "Success",
      "StartTime": 1557593493.222,
      "EndTime": 1557593498.466
    }
  ]
}
```

**Tutorial: Update a maintenance window (AWS CLI)**

This tutorial demonstrates how to use the AWS CLI to update a maintenance window. It also shows you how to update different task types, including those for Systems Manager Run Command, Systems Manager Automation, AWS Lambda, and AWS Step Functions.
The examples in this section use the following Systems Manager actions for updating a maintenance window.

- UpdateMaintenanceWindow
- UpdateMaintenanceWindowTarget
- UpdateMaintenanceWindowTask
- DeregisterTargetFromMaintenanceWindow

For information about using the Systems Manager console to update a maintenance window, see Update or delete a maintenance window (console) (p. 639).

As you follow the steps in this tutorial, replace the values in italicized red text with your own options and IDs. For example, replace the maintenance window ID mw-0c50858d01EXmPLe and the instance ID i-02573cafcfEXmPLe with IDs of resources you create.

To update a maintenance window (AWS CLI)

1. Open the AWS CLI and run the following command to update a target to include a name and a description:

   Linux
   ```bash
   aws ssm update-maintenance-window-target
   --window-id "mw-0c50858d01EXmPLe"
   --window-target-id "e32eeeb2-646c-4f4b-8d1-205fbEXmPLe"
   --name "My-Maintenance-Window-Target"
   --description "Description for my maintenance window target"
   ```

   Windows
   ```bash
   aws ssm update-maintenance-window-target
   --window-id "mw-0c50858d01EXmPLe"
   --window-target-id "e32eeeb2-646c-4f4b-8d1-205fbEXmPLe"
   --name "My-Maintenance-Window-Target"
   --description "Description for my maintenance window target"
   ```

   The system returns information like the following:
   ```json
   {
     "WindowId": "mw-0c50858d01EXmPLe",
     "WindowTargetId": "e32eeeb2-646c-4f4b-8d1-205fbEXmPLe",
     "Targets": [
       {
         "Key": "InstanceIds",
         "Values": [
           "i-02573cafcfEXmPLe"
         ]
       }
     ],
     "Name": "My-Maintenance-Window-Target",
     "Description": "Description for my maintenance window target"
   }
   ```

2. Run the following command to use the replace option to remove the description field and add an additional target. The description field is removed, because the update does not include the field (a null value). Be sure to specify an additional instance that has been configured for use with Systems Manager:

   Linux
   ```bash
   aws ssm update-maintenance-window-target
   --window-id "mw-0c50858d01EXmPLe"
   --window-target-id "e32eeeb2-646c-4f4b-8d1-205fbEXmPLe"
   --replace
   --name "My-Maintenance-Window-Target"
   --description "Description for my maintenance window target"
   ```

   Windows
   ```bash
   aws ssm update-maintenance-window-target
   --window-id "mw-0c50858d01EXmPLe"
   --window-target-id "e32eeeb2-646c-4f4b-8d1-205fbEXmPLe"
   --replace
   --name "My-Maintenance-Window-Target"
   --description "Description for my maintenance window target"
   ```

   The system returns information like the following:
   ```json
   {
     "WindowId": "mw-0c50858d01EXmPLe",
     "WindowTargetId": "e32eeeb2-646c-4f4b-8d1-205fbEXmPLe",
     "Targets": [
       {
         "Key": "InstanceIds",
         "Values": [
           "i-02573cafcfEXmPLe"
         ]
       }
     ],
     "Name": "My-Maintenance-Window-Target",
     "Description": "Description for my maintenance window target"
   }
   ```
### Linux

```bash
aws ssm update-maintenance-window-target \
  --window-id "mw-0c50858d01EXAMPLE" \
  --window-target-id "d208dedf-3f6b-41ff-ace8-8e751EXAMPLE" \
  --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE" \
  --name "My-Maintenance-Window-Target" \
  --replace
```

### Windows

```bash
aws ssm update-maintenance-window-target ^
  --window-id "mw-0c50858d01EXAMPLE" ^
  --window-target-id "d208dedf-3f6b-41ff-ace8-8e751EXAMPLE" ^
  --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE" ^
  --name "My-Maintenance-Window-Target" ^
  --replace
```

The system returns information like the following:

```json
{
  "WindowId": "mw-0c50858d01EXAMPLE",
  "WindowTargetId": "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE",
  "Targets": [
    {
      "Key": "InstanceIds",
      "Values": [
        "i-02573cafcfEXAMPLE",
        "i-0471e04240EXAMPLE"
      ]
    },
  ],
  "Name": "My-Maintenance-Window-Target"
}
```

3. **The start-date option** allows you to delay activation of a maintenance window until a specified future date. The end-date option allows you to set a date and time in the future after which the maintenance window no longer runs. Specify the options in ISO-8601 Extended format.

   Run the following command to specify a date and time range for regularly scheduled maintenance window executions:

   **Linux**

   ```bash
   aws ssm update-maintenance-window \
     --window-id "mw-0c50858d01EXAMPLE" \
     --start-date "2020-10-01T10:10:10Z" \
     --end-date "2020-11-01T10:10:10Z"
   ```

   **Windows**

   ```bash
   aws ssm update-maintenance-window ^
     --window-id "mw-0c50858d01EXAMPLE" ^
     --start-date "2020-10-01T10:10:10Z" ^
     --end-date "2020-11-01T10:10:10Z"
   ```

4. **Run the following command to update a Run Command task.**
Tip
If your target is an EC2 instance for Windows Server, change `df` to `ipconfig`, and AWS-RunShellScript to AWS-RunPowerShellScript in the following command.

### Linux

```bash
aws ssm update-maintenance-window-task
  --window-id "mw-0c50858d01EXAMPLE"
  --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
  --targets "Key=WindowTargetIds,Values=e32eeeb2-646c-4f4b-8ed1-205fbEXAMPLE"
  --service-role-arn "arn:aws:iam::111122233333:role/MaintenanceWindowsRole"
  --task-invocation-parameters "RunCommand={Comment=Revising my Run Command
task,Parameters={commands=df}}"
  --priority 1 --max-concurrency 10 --max-errors 4
  --name "My-Task-Name" --description "A description for my Run Command task"
```

### Windows

```bash
aws ssm update-maintenance-window-task
  --window-id "mw-0c50858d01EXAMPLE"
  --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
  --targets "Key=WindowTargetIds,Values=e32eeeb2-646c-4f4b-8ed1-205fbEXAMPLE"
  --service-role-arn "arn:aws:iam::111122233333:role/MaintenanceWindowsRole"
  --task-invocation-parameters "RunCommand={Comment=Revising my Run Command
task,Parameters={commands=df}}"
  --priority 1 --max-concurrency 10 --max-errors 4
  --name "My-Task-Name" --description "A description for my Run Command task"
```

The system returns information like the following:

```json
{
  "WindowId": "mw-0c50858d01EXAMPLE",
  "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "e32eeeb2-646c-4f4b-8ed1-205fbEXAMPLE"
      ]
    }
  ],
  "TaskArn": "AWS-RunShellScript",
  "ServiceRoleArn": "arn:aws:iam::111122233333:role/MaintenanceWindowsRole",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "RunCommand": {
      "Comment": "Revising my Run Command task",
      "Parameters": {
        "commands": ["df"
      ]
    }
  },
  "Priority": 1,
  "MaxConcurrency": "10",
  "MaxErrors": "4",
  "Name": "My-Task-Name",
```
5. Adapt and run the following command to update a Lambda task.

**Linux**

```bash
aws ssm update-maintenance-window-task
--window-id mw-0c50858d01EXAMPLE
--window-task-id 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
--targets "Key=WindowTargetIds,Values=e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
--service-role-arn "arn:aws:iam::11112223333:role/MaintenanceWindowsRole"
--task-invocation-parameters '{"Lambda":{"Payload":"{"instanceId":\"{{RESOURCE_ID}}\","targetType":\"{{TARGET_TYPE}}\"}}}
--priority 1 --max-concurrency 10 --max-errors 5
--name "New-Lambda-Task-Name"
--description "A description for my Lambda task"
```

**Windows**

```bash
aws ssm update-maintenance-window-task
--window-id mw-0c50858d01EXAMPLE
--window-task-id 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE
--targets "Key=WindowTargetIds,Values=e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
--service-role-arn "arn:aws:iam::11112223333:role/MaintenanceWindowsRole"
--task-invocation-parameters '{"Lambda":{"Payload":"{"instanceId":\"{{RESOURCE_ID}}\","targetType":\"{{TARGET_TYPE}}\"}}}
--priority 1 --max-concurrency 10 --max-errors 5
--name "New-Lambda-Task-Name"
--description "A description for my Lambda task"
```

The system returns information like the following:

```json
{
"WindowId": "mw-0c50858d01EXAMPLE",
"WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
"Targets": [
{
"Key": "WindowTargetIds",
"Values": "e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
}
],
"ServiceRoleArn": "arn:aws:iam::11112223333:role/MaintenanceWindowsRole",
"TaskParameters": {},
"TaskInvocationParameters": {
"Lambda": {
"Payload": "e30=
}
},
"Priority": 1,
"MaxConcurrency": "10",
"MaxErrors": "5",
"Name": "New-Lambda-Task-Name",
"Description": "A description for my Lambda task"
}
```

6. If you are updating an AWS Step Functions task, adapt and run the following command to update its task-invocation-parameters:
Linux

```bash
aws ssm update-maintenance-window-task
   --window-id "mw-0c50858d01EXAMPLE"
   --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
   --targets "Key=WindowTargetIds,Values=e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
   --service-role-arn "arn:aws:iam::11112223333:role/MaintenanceWindowsRole"
   --task-invocation-parameters '{"StepFunctions":{"Input":"{{RESOURCE_ID}}"}}'
   --priority 0 --max-concurrency 10 --max-errors 5
   --name "My-Step-Functions-Task"
   --description "A description for my Step Functions task"
```

Windows

```bash
aws ssm update-maintenance-window-task
   --window-id "mw-0c50858d01EXAMPLE"
   --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
   --targets "Key=WindowTargetIds,Values=e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
   --service-role-arn "arn:aws:iam::11112223333:role/MaintenanceWindowsRole"
   --task-invocation-parameters '{"StepFunctions":{"Input":"{{RESOURCE_ID}}"}}'
   --priority 0 --max-concurrency 10 --max-errors 5
   --name "My-Step-Functions-Task"
   --description "A description for my Step Functions task"
```

The system returns information like the following:

```json
{
"WindowId": "mw-0c50858d01EXAMPLE",
"WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
"Targets": [
   {
      "Key": "WindowTargetIds",
      "Values": [
         "e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
      ]
   }
],
"ServiceRoleArn": "arn:aws:iam::11112223333:role/MaintenanceWindowsRole",
"TaskParameters": {},
"TaskInvocationParameters": {
   "StepFunctions": {
      "Input": "{{RESOURCE_ID}}"
   }
},
"Priority": 0,
"MaxConcurrency": "10",
"MaxErrors": "5",
"Name": "My-Step-Functions-Task",
"Description": "A description for my Step Functions task"
}
```
7. Run the following command to unregister a target from a maintenance window. This example uses the `safe` parameter to determine if the target is referenced by any tasks and therefore safe to unregister:

**Linux**

```
aws ssm deregister-target-from-maintenance-window
  --window-id "mw-0c50858d01EXAMPLE"
  --window-target-id "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
  --safe
```

**Windows**

```
aws ssm deregister-target-from-maintenance-window
  --window-id "mw-0c50858d01EXAMPLE"
  --window-target-id "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
  --safe
```

The system returns information like the following:

An error occurred (TargetInUseException) when calling the DeregisterTargetFromMaintenanceWindow operation:
This Target cannot be deregistered because it is still referenced in Task: 4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE

8. Run the following command to unregister a target from a maintenance window even if the target is referenced by a task. You can force the unregister operation by using the `no-safe` parameter:

**Linux**

```
aws ssm deregister-target-from-maintenance-window
  --window-id "mw-0c50858d01EXAMPLE"
  --window-target-id "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
  --no-safe
```

**Windows**

```
aws ssm deregister-target-from-maintenance-window
  --window-id "mw-0c50858d01EXAMPLE"
  --window-target-id "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
  --no-safe
```

The system returns information like the following:

```
{
  "WindowId": "mw-0c50858d01EXAMPLE",
  "WindowTargetId": "e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE"
}
```

9. Run the following command to update a Run Command task. This example uses a Systems Manager Parameter Store parameter called UpdateLevel, which is formatted as follows: `\'{\{ssm:UpdateLevel\}\}'`

**Linux**

```
aws ssm update-maintenance-window-task
```
Windows

```bash
aws ssm update-maintenance-window-task
   --window-id "mw-0c50858d01EXAMPLE" \
   --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
   --targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE" \
   --task-invocation-parameters "RunCommand={Comment="A comment for my task update",Parameters={UpdateLevel='{{ssm:UpdateLevel}}'}}"
```

The system returns information like the following:

```json
{
   "WindowId": "mw-0c50858d01EXAMPLE",
   "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
   "Targets": [ {
      "Key": "InstanceIds",
      "Values": [ "i-02573cafcfEXAMPLE"
   ]
  ],
   "TaskArn": "AWS-RunShellScript",
   "ServiceRoleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
   "TaskParameters": {},
   "TaskInvocationParameters": { "RunCommand": { "Comment": "A comment for my task update", "Parameters": { "UpdateLevel": [ "{{ssm:UpdateLevel}}"
   ]
   } },
   "Priority": 10,
   "MaxConcurrent": "1",
   "MaxErrors": "1"
}
```

10. Run the following command to update an Automation task to specify WINDOW_ID and WINDOW_TASK_ID parameters for the task-invocation-parameters parameter:

Linux

```bash
aws ssm update-maintenance-window-task
   --window-id "mw-0c50858d01EXAMPLE" \
   --window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" \
   --targets "Key=WindowTargetIds,Values=e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE" \
   --task-arn "AutoTestDoc" \
   --service-role-arn arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM
```
Windows

aws ssm update-maintenance-window-task ^
--window-id "mw-0c50858d01EXAMPLE" ^
--window-task-id "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE" ^
--targets "Key=WindowTargetIds,Values=e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE" ^
--task-arn "AutoTestDoc" ^
--service-role-arn arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM ^
--task-invocation-parameters
"Automation={Parameters={instanceId='{{RESOURCE_ID}}',initiator='{{WINDOW_ID}}.Task-{{WINDOW_TASK_ID}}'}}" ^
--priority 3 --max-concurrency 10 --max-errors 5

The system returns information like the following:

```json
{
  "WindowId": "mw-0c50858d01EXAMPLE",
  "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE",
  "Targets": [
    {
      "Key": "WindowTargetIds",
      "Values": [
        "e32eeb2-646c-4f4b-8ed1-205fbEXAMPLE"
      ]
    }
  ],
  "TaskArn": "AutoTestDoc",
  "ServiceRoleArn": "arn:aws:iam::111122223333:role/aws-service-role/ssm.amazonaws.com/AWSServiceRoleForAmazonSSM",
  "TaskParameters": {},
  "TaskInvocationParameters": {
    "Automation": {
      "Parameters": {
        "multi": [
          "{{WINDOW_TASK_ID}}"
        ],
        "single": [
          "{{WINDOW_ID}}"
        ]
      }
    },
    "Priority": 0,
    "MaxConcurrency": "10",
    "MaxErrors": "5",
    "Name": "My-Automation-Task",
    "Description": "A description for my Automation task"
  }
}
```

# Tutorial: Delete a maintenance window (AWS CLI)

To delete a maintenance window you created in these tutorials, run the following command:

```bash
aws ssm delete-maintenance-window-task
--window-id "mw-0c50858d01EXAMPLE"
```

The system returns information like the following:

```json
{ "WindowId": "mw-0c50858d01EXAMPLE" }```
aws ssm delete-maintenance-window --window-id "mw-0c50858d01EXAMPLE"

The system returns information like the following:

```
{
  "WindowId":"mw-0c50858d01EXAMPLE"
}
```

## Maintenance window walkthroughs

The walkthroughs in this section show you how to create an AWS Systems Manager maintenance window using either the AWS Command Line Interface or the AWS Systems Manager Console. The maintenance window that you create updates SSM Agent on managed instances.

### Contents

- Walkthrough: Create a maintenance window to update SSM Agent (AWS CLI) (p. 685)
- Walkthrough: Create a maintenance window to update SSM Agent (console) (p. 690)

You can also view sample commands in the Systems Manager AWS CLI Reference.

### Walkthrough: Create a maintenance window to update SSM Agent (AWS CLI)

The following walkthrough shows you how to use the AWS CLI to create an AWS Systems Manager maintenance window. The walkthrough also describes how to register your managed instances as targets and register a Run Command task to update SSM Agent.

#### Before You Begin

Before you complete the following procedure, you must either have administrator privileges on the instances you want to configure or you must have been granted the appropriate permissions in AWS Identity and Access Management (IAM). Additionally, verify that you have at least one running EC2 instance for Linux or Windows Server that is configured for Systems Manager. For more information, see Systems Manager prerequisites (p. 14).

#### Topics

- Step 1: Get started (p. 685)
- Step 2: Create the maintenance window (p. 686)
- Step 3: Register maintenance window targets (AWS CLI) (p. 687)
- Step 4: Register a Run Command task for the maintenance window to update SSM Agent (p. 689)

#### Step 1: Get started

**To run commands using the AWS CLI**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Verify that an instance is ready to be registered as a target for a maintenance window.

   Run the following command to view which instances are online.
aws ssm describe-instance-information --query "InstanceInformationList[*]"

Run the following command to view details about a particular instance.

aws ssm describe-instance-information --instance-information-filter-list
key=InstanceIds,valueSet=instance-id

**Step 2: Create the maintenance window**

Use the following procedure to create a maintenance window and specify its basic options, such as schedule and duration.

**Create a maintenance window (AWS CLI)**

1. Open the AWS CLI and run the following commands to create a maintenance window that runs weekly on Sundays at 02:00, in the United States Pacific time zone, with a one hour cutoff:

   **Linux**
   ```bash
   aws ssm create-maintenance-window
   --name "My-First-Maintenance-Window"
   --schedule "cron(0 2 ? * SUN *)"
   --duration 2
   --schedule-timezone "America/Los_Angeles"
   --cutoff 1
   --no-allow-unassociated-targets
   ```

   **Windows**
   ```bash
   aws ssm create-maintenance-window
   --name "My-First-Maintenance-Window"
   --schedule "cron(0 2 ? * SUN *)"
   --duration 2
   --schedule-timezone "America/Los_Angeles"
   --cutoff 1
   --no-allow-unassociated-targets
   ```

   For information about creating cron expressions for the `schedule` parameter, see [Reference: Cron and rate expressions for Systems Manager (p. 1189)](#).

   For an explanation of how the various schedule-related options for maintenance windows relate to one another, see [Reference: Maintenance window scheduling and active period options (p. 693)](#).

   The system returns information like the following:

   ```json
   {
     "WindowId":"mw-0c50858d01EXAMPLE"
   }
   ```

2. To list this and any other maintenance windows created in your AWS account in your current AWS Region, run the following command:

   ```bash
   aws ssm describe-maintenance-windows
   ```
The system returns information like the following:

```
{
    "WindowIdentities": [
        {
            "Cutoff": 1,
            "Name": "My-First-Maintenance-Window",
            "NextExecutionTime": "2019-02-03T02:00-08:00",
            "Enabled": true,
            "WindowId": "mw-0c50858d01EXAMPLE",
            "Duration": 2
        }
    ]
}
```

**Step 3: Register maintenance window targets (AWS CLI)**

Use the following procedure to register a target with your maintenance window created in Step 2. By registering a target, you specify which instances to update.

**To register maintenance window targets (AWS CLI)**

1. Run the following command:

   **Linux**
   ```bash
   aws ssm register-target-with-maintenance-window \
   --window-id "mw-0c50858d01EXAMPLE" \
   --target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE" \
   --resource-type "INSTANCE"
   ```

   **Windows**
   ```bash
   aws ssm register-target-with-maintenance-window ^
   --window-id "mw-0c50858d01EXAMPLE" ^
   --target "Key=InstanceIds,Values=i-02573cafcfEXAMPLE" ^
   --resource-type "INSTANCE"
   ```

   The system returns information like the following, which includes a maintenance window target ID. Copy or note the WindowTargetId value. You must specify this ID in the next step to register a task for this maintenance window.

   ```json
   {
     "WindowTargetId": "1a2b3c4d-1a2b-1a2b-1a2b-1a2b3c4d-1a2"
   }
   ```

**Alternative commands**

Use the following command to register multiple managed instances:

**Linux**
```bash
aws ssm register-target-with-maintenance-window \
--window-id "mw-0c50858d01EXAMPLE" \
--targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE, i-0471e04240EXAMPLE"
```
Use the following command to register instances by using Amazon EC2 tags. For example:

**Windows**

```bash
aws ssm register-target-with-maintenance-window
--window-id "mw-0c50858d01EXAMPLE"
--targets "Key=InstanceIds,Values=i-02573cafcfEXAMPLE,i-0471e04240EXAMPLE"
--resource-type "INSTANCE"
```

**Linux**

```bash
aws ssm register-target-with-maintenance-window
--window-id "mw-0c50858d01EXAMPLE"
--targets "Key=tag:Environment,Values=Prod" "Key=tag:Role,Values=Web"
--resource-type "INSTANCE"
```

2. Run the following command to display the targets for a maintenance window:

```bash
aws ssm describe-maintenance-window-targets --window-id "mw-0c50858d01EXAMPLE"
```

The system returns information like the following:

```json
{
    "Targets": [
        {
            "ResourceType": "INSTANCE",
            "WindowId": "mw-0c50858d01EXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "i-02573cafcfEXAMPLE"
                    ],
                    "Key": "InstanceIds"
                }
            ],
            "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
        },
        {
            "ResourceType": "INSTANCE",
            "WindowId": "mw-0c50858d01EXAMPLE",
            "Targets": [
                {
                    "Values": [
                        "Prod"
                    ],
                    "Key": "tag:Environment"
                },
                {
                    "Values": [
                        "Web"
                    ],
                    "Key": "tag:Role"
                }
            ],
            "WindowTargetId": "e32eecb2-646c-4f4b-8ed1-205fbEXAMPLE"
        }
    ]
}
```
Step 4: Register a Run Command task for the maintenance window to update SSM Agent

Use the following procedure to register a Run Command task for the maintenance window you created in Step 2. The Run Command task updates SSM Agent on the registered targets.

To register a Run Command task for a maintenance window to update SSM Agent (AWS CLI)

1. Run the following command to register a Run Command task for the maintenance window using the WindowTargetId value in Step 3. The task updates SSM Agent by using the \texttt{AWS-UpdateSSMAgent} document.

   \textbf{Linux}

   \begin{verbatim}
aws ssm register-task-with-maintenance-window \
   --window-id "mw-0c50858d01EXAMPLE" \
   --task-arn "AWS-UpdateSSMAgent" \
   --name "UpdateSSMAgent" \
   --targets "Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE" \
   --service-role-arn "arn:aws:iam::1122334455:role/MW-Role" \
   --task-type "RUN_COMMAND" \
   --max-concurrency 1 --max-errors 1 --priority 10
   \end{verbatim}

   \textbf{Windows}

   \begin{verbatim}
aws ssm register-task-with-maintenance-window ^
   --window-id "mw-0c50858d01EXAMPLE" ^
   --task-arn "AWS-UpdateSSMAgent" ^
   --name "UpdateSSMAgent" ^
   --targets "Key=WindowTargetIds,Values=e32eebc2-646c-4f4b-8ed1-205fbEXAMPLE" ^
   --service-role-arn "arn:aws:iam::1122334455:role/MW-Role" ^
   --task-type "RUN_COMMAND" ^
   --max-concurrency 1 --max-errors 1 --priority 10
   \end{verbatim}

\textbf{Note}

If the targets you registered in the preceding step are Windows Server 2012 R2 or earlier, you must use the \texttt{AWS-UpdateEC2Config} document.

The system returns information like the following:

\begin{verbatim}
{
   "WindowTaskId": "4f7ca192-7e9a-40fe-9192-5cb15EXAMPLE"
}
\end{verbatim}

2. Run the following command to list all registered tasks for a maintenance window.

\begin{verbatim}
aws ssm describe-maintenance-window-tasks --window-id "mw-0c50858d01EXAMPLE"
\end{verbatim}
Walkthrough: Create a maintenance window to update SSM Agent (console)

The following walkthrough shows you how to use the AWS Systems Manager console to create an AWS Systems Manager maintenance window. The walkthrough also describes how to register your managed instances as targets and register a Run Command task to update SSM Agent.

Before You Begin

Before you complete the following procedure, you must either have administrator privileges on the instances you want to configure or you must have been granted the appropriate permissions in AWS Identity and Access Management (IAM). Additionally, verify that you have at least one running EC2 instance for Linux or Windows Server that is configured for Systems Manager. For more information, see Systems Manager prerequisites (p. 14).

Topics

- Step 1: Create the maintenance window (console) (p. 690)
- Step 2: Register maintenance window targets (console) (p. 691)
- Step 3: Register a Run Command task for the maintenance window to update SSM Agent (console) (p. 692)

Step 1: Create the maintenance window (console)

To create a maintenance window (console)

2. In the navigation pane, choose Maintenance Windows.
3. Choose Create maintenance window.
4. For **Name**, enter a descriptive name to help you identify this maintenance window as a test maintenance window.

5. For **Description**, enter a description.

6. Choose **Allow unregistered targets** if you want to allow a maintenance window task to run on managed instances, even if you have not registered those instances as targets. If you choose this option, then you can choose the unregistered instances (by instance ID) when you register a task with the maintenance window.

   If you don't choose this option, then you must choose previously-registered targets when you register a task with the maintenance window.

7. Specify a schedule for the maintenance window by using one of the three scheduling options.

   For information about building cron/rate expressions, see Reference: Cron and rate expressions for Systems Manager (p. 1189).

8. For **Duration**, enter the number of hours the maintenance window should run.

9. For **Stop initiating tasks**, enter the number of hours before the end of the maintenance window that the system should stop scheduling new tasks to run.

10. (Optional) For **Start date (optional)**, specify a date and time, in ISO-8601 Extended format, for when you want the maintenance window to become active. This allows you to delay activation of the maintenance window until the specified future date.

11. (Optional) For **End date (optional)**, specify a date and time, in ISO-8601 Extended format, for when you want the maintenance window to become inactive. This allows you to set a date and time in the future after which the maintenance window no longer runs.

12. (Optional) For **Time zone (optional)**, specify the time zone to base scheduled maintenance window executions on, in Internet Assigned Numbers Authority (IANA) format. For example: "America/Los_Angeles", "etc/UTC", or "Asia/Seoul".

   For more information about valid formats, see the Time Zone Database on the IANA website.

13. Choose **Create maintenance window**. The system returns you to the maintenance window page. The maintenance window you just created is in the Enabled state.

**Step 2: Register maintenance window targets (console)**

Use the following procedure to register a target with the maintenance window you created in Step 1. By registering a target, you specify which instances to update.

**To assign targets to a maintenance window (console)**

1. In the list of maintenance windows, choose the maintenance window you just created.

2. Choose **Actions**, and then choose **Register targets**.

3. For **Target Name**, enter a name for the target.

4. For **Description**, enter a description.

5. (Optional) For **Owner information**, specify your name or work alias. Owner information is included in any Amazon CloudWatch Events raised while running tasks for these targets in this maintenance window.

   For information about using CloudWatch Events to monitor Systems Manager events, see Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147).

6. In the **Select targets by** section, choose **Specifying Tags** to target instances by using Amazon EC2 tags that you previously assigned to the instances. Choose **Manually Selecting Instances** to choose individual instances according to their instance IDs.
Note
If you don’t see the instances that you want to target, verify that those instances are configured for Systems Manager. For more information, see Setting up AWS Systems Manager (p. 25).

7. Choose Register target.

Step 3: Register a Run Command task for the maintenance window to update SSM Agent (console)

Use the following procedure to register a Run Command task for the maintenance window you created in Step 1. The Run Command task updates SSM Agent on the registered targets.

To assign tasks to a maintenance window (console)

1. In the list of maintenance windows, choose the maintenance window you just created.
2. Choose Actions, and then choose Register Run Command task.
3. For Name, enter a name for the task, such as UpdateSSMAgent.
4. For Description, enter a description.
5. For Document, choose the SSM Command document AWS-UpdateSSMAgent.

Note
If the targets you registered in the preceding step are Windows Server 2012 R2 or earlier, you must use the AWS-UpdateEC2Config document.

6. For Task priority, specify a priority for this task. Zero (0) is the highest priority. Tasks in a maintenance window are scheduled in priority order with tasks that have the same priority scheduled in parallel.

7. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

Note
If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.

8. (Optional) For Rate control:
   - For Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.

Note
If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

   - For Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

9. For IAM service role, choose one of the following options to provide permissions for Systems Manager to run tasks on your target instances:
   - Create and use a service-linked role for Systems Manager

Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role. Additionally, AWS automatically defines and sets the permissions of service-linked roles, depending on the actions that the linked service performs on your behalf.
Note
If a service-linked role has already been created for your account, choose **Use the service-linked role for Systems Manager.**

- **Use a custom service role**

You can create a custom service role for maintenance window tasks if you want to use stricter permissions than those provided by the service-linked role.

If you need to create a custom service role, see one of the following topics:

- Control access to maintenance windows (console) (p. 622)
- Control access to maintenance windows (AWS CLI) (p. 626)
- Control access to maintenance windows (Tools for Windows PowerShell) (p. 630)

To help you decide whether to use a custom service role or the Systems Manager service-linked role with a maintenance window task, see **Should I use a service-linked role or a custom service role to run maintenance window tasks?** (p. 622).

10. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Type the bucket and prefix (folder) names in the boxes.

**Note**
The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Control access to maintenance windows (console) (p. 622).

To stream the output to a CloudWatch Logs log group, select the **CloudWatch output** box. Type the log group name in the box.

11. In the **SNS notifications** section, you can optionally enable Systems Manager to send notifications about command statuses using Amazon SNS. If you choose to enable this option, you need to specify the following:

   a. The IAM role to trigger Amazon SNS notifications.
   b. The Amazon SNS topic to be used.
   c. The specific event types about which you want to be notified.
   d. The notification type that you want to receive when the status of a command changes. For commands sent to multiple instances, choose **Invocation** to receive notification on an invocation (per-instance) basis when the status of each invocation changes.

12. In the **Input Parameters** section, you can optionally provide a specific version of SSM Agent to install, or you can allow SSM Agent service to be downgraded to an earlier version. However, for this walkthrough we don’t provide a version. Therefore, SSM Agent is be updated to the latest version.

13. Choose **Register Run Command task**.

**Reference: Maintenance window scheduling and active period options**

When you create a maintenance window, you must specify how often the maintenance window runs by using a **Cron or rate expression** (p. 1189). Optionally, you can specify a date range during which the maintenance window can run on its regular schedule, as well as a time zone on which to base that regular schedule.
Be aware, however, that the time zone option and the start date/end date options do not influence each other. Any start date and end date times that you specify (with or without an offset for your time zone) determine only the valid period during which the maintenance window can run on its schedule. A time zone option determines the international time zone that the maintenance window schedule is based on during its valid period.

**Note**
You specify start and end dates in ISO-8601 timestamp format. For example:

```
2019-04-07T14:29:00-08:00
```

You specify time zones in Internet Assigned Numbers Authority (IANA) format. For example:

```
America/Chicago, Europe/Berlin or Asia/Tokyo
```

**Examples**

- Example 1: Specify a maintenance window start date (p. 694)
- Example 2: Specify a maintenance window start date and end date (p. 694)
- Example 3: Create a maintenance window that runs only once (p. 695)

**Example 1: Specify a maintenance window start date**

Say that you use the AWS CLI to create a maintenance window with the following options:

- `--start-date 2019-01-01T00:00:00-05:00`
- `--schedule-timezone "America/Los_Angeles"
- `--schedule "cron(0 09 ? * FRI *)"

For example:

```
aws ssm create-maintenance-window --name "My-LAX-Maintenance-Window" --start-date 2019-01-01T00:00:00-08:00 --schedule-timezone "America/Los_Angeles" --schedule "cron(0 09 ? * FRI *)"
```

This means that the maintenance window won't run until after its specified start date and time, which is at Midnight US Eastern Time on Tuesday, January 1, 2019. (This time zone is five hours behind UTC time.) Taken together, the `--schedule-timezone` and `--schedule` values mean that the maintenance window runs at 9 AM every Friday in the US Pacific Time Zone (represented by 'America/Los_Angeles' in IANA format). The first execution in the enabled period will be on Friday, January 4th, 2019, at 9 AM US Pacific Time.

**Example 2: Specify a maintenance window start date and end date**

Suppose that next you create a maintenance window with these options:

- `--start-date 2019-01-01T00:03:15+09:00`
- `--end-date 2019-06-30T00:06:15+09:00`
- `--schedule-timezone "Asia/Tokyo"
- `--schedule "rate(7 days)"

For example:

```
aws ssm create-maintenance-window --name "My-NRT-Maintenance-Window" --start-date 2019-01-01T00:03:15+09:00 --end-date 2019-06-30T00:06:15+09:00 --schedule-timezone "Asia/Tokyo" --schedule "rate(7 days)"
```
The enabled period for this maintenance window begins at 3:15 AM Japan Standard Time on January 1, 2019. The valid period for this maintenance window ends at 6:15 AM Japan Standard Time on Sunday, June 30, 2019. (This time zone is nine hours ahead of UTC time.) Taken together, the --schedule-timezone and --schedule values mean that the maintenance window runs at 3:15 AM every Tuesday in the Japan Standard Time Zone (represented by "Asia/Tokyo" in IANA format). This is because the maintenance window runs every seven days, and it becomes active at 3:15 AM on Tuesday, January 1st. The last execution is at 3:15 AM Japan Standard Time on Tuesday, June 25, 2019. This is the last Tuesday before the enabled maintenance window period ends five days later.

**Example 3: Create a maintenance window that runs only once**

Now you create a maintenance window with this option:

- `--schedule "at(2020-07-07T15:55:00)"

For example:

```bash
aws ssm create-maintenance-window --name "My-One-Time-Maintenance-Window" --schedule "at(2020-07-07T15:55:00)" --duration 5 --cutoff 2 --allow-unassociated-targets
```

This maintenance window runs just once, at 3:55 PM UTC time on July 7, 2020. The maintenance window is enabled to run up to five hours, as needed, but new tasks are prevented from starting two hours before the end of the maintenance window period.

**Related Content**

- Reference: Cron and rate expressions for Systems Manager (p. 1189)
- Create a maintenance window (console) (p. 634)
- Tutorial: Create and configure a maintenance window (AWS CLI) (p. 642)
- CreateMaintenanceWindow in the AWS Systems Manager API Reference
- create-maintenance-window in the AWS Systems Manager section of the AWS CLI Command Reference
- Time Zone Database on the IANA website
AWS Systems Manager Instances & Nodes

AWS Systems Manager provides the following capabilities for managing your EC2 instances, your on-premises servers and virtual machines (VMs) in your hybrid environment, and other types of AWS resources (nodes).

Topics
- AWS Systems Manager Configuration Compliance (p. 696)
- AWS Systems Manager Inventory (p. 704)
- AWS Systems Manager Managed Instances (p. 758)
- AWS Systems Manager hybrid activations (p. 769)
- AWS Systems Manager Session Manager (p. 770)
- AWS Systems Manager Run Command (p. 829)
- AWS Systems Manager State Manager (p. 864)
- AWS Systems Manager Patch Manager (p. 911)
- AWS Systems Manager Distributor (p. 999)

AWS Systems Manager Configuration Compliance

You can use AWS Systems Manager Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren't compliant. By default, Configuration Compliance displays current compliance data about Systems Manager Patch Manager patching and Systems Manager State Manager associations. Systems Manager Compliance offers the following additional benefits and features:

- View compliance history and change tracking for Patch Manager patching data and State Manager associations by using AWS Config.
- Customize Systems Manager Compliance to create your own compliance types based on your IT or business requirements.
- Remediate issues by using Systems Manager Run Command, State Manager, or Amazon CloudWatch Events.
- Port data to Amazon Athena and Amazon QuickSight to generate fleet-wide reports.

Configuration Compliance is offered at no additional charge. You only pay for the AWS resources that you use.

Note
Systems Manager integrates with Chef InSpec. InSpec is an open-source, runtime framework that enables you to create human-readable profiles on GitHub or Amazon S3. Then you can use Systems Manager to run compliance scans and view compliant and noncompliant instances. For more information, see Using Chef InSpec profiles with Systems Manager Compliance (p. 113).
Getting started with Configuration Compliance

To get started with Configuration Compliance, complete the following tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>For more information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Compliance works with Patch Manager patch data, State Manager associations, and custom compliance types on Systems Manager managed instances. Verify that your EC2 instances and hybrid machines (on-premises instances and virtual machines (VMs) are configured as managed instances by verifying Systems Manager prerequisites.</td>
<td>Systems Manager prerequisites (p. 14)</td>
</tr>
<tr>
<td>Update SSM Agent on your managed instances to the latest version.</td>
<td>Working with SSM Agent (p. 63)</td>
</tr>
<tr>
<td>If you plan to monitor patch compliance, verify that you’ve configured Systems Manager Patch Manager. You must perform patching operations by using Patch Manager before Configuration Compliance can display patch compliance data.</td>
<td>AWS Systems Manager Patch Manager (p. 911)</td>
</tr>
<tr>
<td>If you plan to monitor association compliance, verify that you’ve created State Manager associations. You must create associations before Configuration Compliance can display association compliance data.</td>
<td>AWS Systems Manager State Manager (p. 864)</td>
</tr>
<tr>
<td>(Optional) Configure the system to view compliance history and change tracking.</td>
<td>Viewing compliance configuration history and change tracking (p. 701)</td>
</tr>
<tr>
<td>(Optional) Create custom compliance types.</td>
<td>Configuration Compliance walkthrough (AWS CLI) (p. 703)</td>
</tr>
<tr>
<td>(Optional) Create a Resource Data Sync to aggregate all compliance data in a target S3 bucket.</td>
<td>Creating a Resource Data Sync for Configuration Compliance (p. 697)</td>
</tr>
</tbody>
</table>

Creating a Resource Data Sync for Configuration Compliance

You can use Systems Manager Resource Data Sync to send compliance data from all of your managed instances to a target S3 bucket. When you create the sync, you can specify managed instances from
multiple AWS accounts, AWS Regions, and your on-premises hybrid environment. Resource Data Sync then automatically updates the centralized data when new compliance data is collected. With all compliance data stored in a target S3 bucket, you can use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data. Configuring Resource Data Sync for configuration compliance is a one-time operation.

The following graphic shows how Resource Data Sync aggregates all data from different accounts, Regions, and your hybrid environment to a central repository.

Use the following procedure to create a Resource Data Sync for Configuration Compliance by using the Amazon EC2 console.

To create and configure an S3 bucket for Resource Data Sync (console)

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket to store your aggregated Inventory data. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. Open the bucket, choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace Bucket-Name and Account-ID with the name of the S3 bucket you created and a valid AWS account ID. Optionally, replace Bucket-Prefix with the name of an Amazon S3 prefix (subdirectory). If you didn't create a prefix, remove Bucket-Prefix/ from the ARN in the policy.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SSMBucketPermissionsCheck",
      "Effect": "Allow",
      "Principal": {
        "Service": "ssm.amazonaws.com"
      },
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::Bucket-Name"
    }
  ]
}
```
To create a Resource Data Sync

2. In the navigation pane, choose **Managed Instances**.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Managed Instances**.

3. Choose **Configure Inventory, Resource Data Syncs**, and then choose **Create resource data sync**.
4. In the **Sync name** field, type a name for the sync configuration.
5. In the **Bucket name** field, type the name of the Amazon S3 bucket you created at the start of this procedure.
6. (Optional) In the **Bucket prefix** field, type the name of an S3 bucket prefix (subdirectory).
7. In the **Bucket region** field, choose **This region** if the S3 bucket you created is located in the current AWS Region. If the bucket is located in a different AWS Region, choose **Another region**, and type the name of the Region.

   **Note**
   If the sync and the target S3 bucket are located in different regions, you may be subject to data transfer pricing. For more information, see Amazon S3 Pricing.

8. Choose **Create**.

**Working with Configuration Compliance**

Configuration Compliance collects and reports data about the status of Patch Manager patching, State Manager associations, and custom compliance types. This section includes details about each of these compliance types and how to view Systems Manager compliance data. This section also includes information about how to view compliance history and change tracking.

**Note**
Systems Manager integrates with Chef InSpec. InSpec is an open-source, runtime framework that enables you to create human-readable profiles on GitHub or Amazon S3. Then you can use Systems Manager to run compliance scans and view compliant and noncompliant instances. For more information, see Using Chef InSpec profiles with Systems Manager Compliance (p. 113).
About patch compliance

After you use Systems Manager Patch Manager to install patches on your instances, compliance status information is immediately available to you in the console or in response to AWS CLI commands or corresponding Systems Manager API actions.

For information about patch compliance status values, see About patch compliance status values (p. 946).

About State Manager association compliance

After you create one or more State Manager associations, compliance status information is immediately available to you in the console or in response to AWS CLI commands or corresponding Systems Manager API actions. For associations, Configuration Compliance shows statuses of Compliant or Non-compliant and the severity level assigned to the association, such as Critical or Medium.

About custom compliance

You can assign compliance metadata to a managed instance. This metadata can then be aggregated with other compliance data for compliance reporting purposes. For example, say that your business runs versions 2.0, 3.0, and 4.0 of software X on your managed instances. The company wants to standardize on version 4.0, meaning that instances running versions 2.0 and 3.0 are non-compliant. You can use the PutComplianceItems API action to explicitly note which managed instances are running older versions of software X. Currently you can only assign compliance metadata by using the AWS CLI, AWS Tools for Windows PowerShell, or the SDKs. The following CLI sample command assigns compliance metadata to a managed instance and specifies the compliance type in the required format Custom:

```bash
aws ssm put-compliance-items --resource-id i-1234567890abcdef0 --resource-type ManagedInstance --compliance-type Custom:SoftwareXCheck --execution-summary ExecutionTime=AnyStringToDenoteTimeOrDate --items
Id=Version2.0,Title=SoftwareXVersion,Severity=CRITICAL,Status=NON_COMPLIANT
```

Compliance managers can then view summaries or create reports about which instances are or aren't compliant. You can assign a maximum of 10 different custom compliance types to an instance.

For an example of how to create a custom compliance type and view compliance data, see Configuration Compliance walkthrough (AWS CLI) (p. 703).

Viewing current compliance data

This section describes how to view compliance data in the AWS Systems Manager console and by using the AWS CLI. For information about how to view patch and association compliance history and change tracking, see Viewing compliance configuration history and change tracking (p. 701).

Topics

- Viewing current compliance data (console) (p. 701)
- Viewing current compliance data (AWS CLI) (p. 701)
**Viewing current compliance data (console)**

Use the following procedure to view compliance data in the Systems Manager console.

**To view current compliance reports in the Systems Manager console**

2. In the navigation pane, choose Compliance.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Compliance in the navigation pane.
3. In the Details overview for resources area, choose the ID of an instance.
4. On the Instance ID details page, select the Configuration compliance tab to view its detailed configuration compliance report.

**Note**

For information about fixing compliance issues, see Remediating compliance issues (p. 702).

**Viewing current compliance data (AWS CLI)**

You can view summaries of compliance data for patching, associations, and custom compliance types in the in the AWS CLI by using the following AWS CLI commands.

- **list-compliance-summaries**
  Returns a summary count of compliant and non-compliant association statuses according to the filter you specify. (API: ListComplianceSummaries)

- **list-resource-compliance-summaries**
  Returns a resource-level summary count. The summary includes information about compliant and non-compliant statuses and detailed compliance-item severity counts, according to the filter criteria you specify. (API: ListResourceComplianceSummaries)

You can view additional compliance data for patching by using the following AWS CLI commands.

- **describe-patch-group-state**
  Returns high-level aggregated patch compliance state for a patch group. (API: DescribePatchGroupState)

- **describe-instance-patch-states-for-patch-group**
  Returns the high-level patch state for the instances in the specified patch group. (API: DescribeInstancePatchStatesForPatchGroup)

**Note**

For an illustration of how to configure patching and view patch compliance details by using the AWS CLI, see Walkthrough: Patch a server environment (command line) (p. 991).

**Viewing compliance configuration history and change tracking**

Systems Manager Configuration Compliance displays current patching and association compliance data for your managed instances. You can view patching and association compliance history and change tracking by using AWS Config. AWS Config provides a detailed view of the configuration of AWS
resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time. To view patching and association compliance history and change tracking, you must enable the following resources in AWS Config:

- SSM:PatchCompliance
- SSM:AssociationCompliance

For information about how to choose and configure these specific resources in AWS Config, see Selecting Which Resources AWS Config Records in the AWS Config Developer Guide.

Note
For information about AWS Config pricing, see Pricing.

Remediating compliance issues

You can quickly remediate patch and association compliance issues by using Systems Manager Run Command. You can target either instance IDs or Amazon EC2 tags and run the AWS-RunPatchBaseline document or the AWS-RefreshAssociation document. If refreshing the association or re-running the patch baseline fails to resolve the compliance issue, then you need to investigate your associations, patch baselines, or instance configurations to understand why the Run Command executions did not resolve the problem.

For more information about patching, see AWS Systems Manager Patch Manager (p. 911) and About the SSM document AWS-RunPatchBaseline (p. 938).

For more information about associations, see Working with associations in Systems Manager (p. 867).

For more information about running a command, see Running commands using Systems Manager Run Command (p. 833).

Specify Configuration Compliance as the target of a CloudWatch Events event

You can also configure CloudWatch Events to perform an action in response to Configuration Compliance events. For example, if one or more instances fail to install Critical patch updates or run an association that installs anti-virus software, then you can configure CloudWatch to run the AWS-RunPatchBaseline document or the AWS-RefreshAssociation document when the Configuration Compliance event occurs.

Use the following procedure to configure Configuration Compliance as the target of a CloudWatch event.

To configure Configuration Compliance as the target of a CloudWatch event (console)

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Choose Event Pattern. Event Pattern lets you build a rule that generates events for specific actions in AWS services.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM)
5. In the Event Type field, choose Configuration Compliance.
6. Choose Add target.
7. In the Select target type list, choose SSM Run Command.
8. In the Document list, choose an SSM document to run when your target is invoked. For example, choose AWS-RunPatchBaseline for a non-compliant patch event, or choose AWS-RefreshAssociation for a non-compliant association event.
9. Specify information for the remaining fields and parameters.
Note
Required fields and parameters have an asterisk (*) next to the name. To create a target, you must specify a value for each required parameter or field. If you don't, the system creates the rule, but the rule won't be run.

10. Choose Configure details and complete the wizard.

Configuration Compliance walkthrough (AWS CLI)

The following procedure walks you through the process of using the PutComplianceItems API action to assign custom compliance metadata to a resource. You can also use this API action to manually assign patch or association compliance metadata to an instance, as shown in the following walkthrough. For more information about custom compliance, see About custom compliance (p. 700).

To assign custom compliance metadata to a managed instance (AWS CLI)

1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to assign custom compliance metadata to an instance. Currently the only supported resource type is ManagedInstance.

   ```bash
   aws ssm put-compliance-items --resource-id Instance ID --resource-type ManagedInstance --compliance-type Custom --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
   ```

3. Repeat the previous step to assign additional custom compliance metadata to one or more instances. You can also manually assign patch or association compliance metadata to managed instances by using the following commands:

   Association compliance metadata

   ```bash
   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\}"
   ```

   Patch compliance metadata

   ```bash
   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\}"
   ```

4. Run the following command to view a list of compliance items for a specific managed instance. Use filters to drill down into specific compliance data.

   ```bash
   aws ssm list-compliance-items --resource-ids Instance ID --resource-types ManagedInstance --filters One or more filters.
   ```
The following examples show you how to use this command with filters.

```
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-a88cc6c46d95 Key=Severity,Values=UNSPECIFIED
```

```
aws ssm list-resource-compliance-summaries --filters
Key=OverallSeverity,Values=UNSPECIFIED
```

5. Run the following command to view a summary of compliance statuses. Use filters to drill down into specific compliance data.

```
aws ssm list-resource-compliance-summaries --filters
```

The following examples show you how to use this command with filters.

```
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
```

6. Run the following command to view a summary count of compliant and non-compliant resources for a compliance type. Use filters to drill down into specific compliance data.

```
aws ssm list-compliance-summaries --filters
```

The following examples show you how to use this command with filters.

```
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 Inventory**

AWS Systems Manager Inventory provides visibility into your Amazon EC2 and on-premises computing environment. You can use Inventory to collect metadata from your managed instances. You can store this metadata in a central Amazon Simple Storage Service (Amazon S3) bucket, and then use built-in tools to query the data and quickly determine which instances are running the software and configurations required by your software policy, and which instances need to be updated. You can configure Inventory on all of your managed instances by using a one-click procedure. You can also configure and view inventory data from multiple AWS Regions and accounts.
If the pre-configured metadata types collected by Systems Manager Inventory don't meet your needs, then you can create custom inventory. Custom inventory is simply a JSON file with information that you provide and add to the managed instance in a specific directory. When Systems Manager Inventory collects data, it captures this custom inventory data. For example, if you run a large datacenter, you can specify the rack location of each of your servers as custom inventory. You can then view the rack space data when you view other inventory data.

**Important**

Systems Manager Inventory collects only metadata from your managed instances. Inventory does not access proprietary information or data.

The following table lists the types of metadata that you can collect with Systems Manager Inventory. The table also lists the instances you can collect inventory information from and the collection intervals you can specify.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata types</td>
<td>You can configure Inventory to collect the following types of metadata:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Applications</strong>: Application names, publishers, versions, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>AWS components</strong>: EC2 driver, agents, versions, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Files</strong>: Name, size, version, installed date, modification and last accessed times, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Network configuration</strong>: IP address, MAC address, DNS, gateway, subnet mask, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Windows updates</strong>: Hotfix ID, installed by, installed date, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Instance details</strong>: System name, operating systems (OS) name, OS version, last boot, DNS, domain, work group, OS architecture, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Services</strong>: Name, display name, status, dependent services, service type, start type, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Tags</strong>: Tags assigned to your instances.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Windows Registry</strong>: Registry key path, value name, value type, and value.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Windows roles</strong>: Name, display name, path, feature type, installed state, etc.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Custom inventory</strong>: Metadata that was assigned to a managed instance as described in Working with custom inventory (p. 736).</td>
</tr>
</tbody>
</table>

**Note**

To view a list of all metadata collected by Inventory, see Metadata collected by inventory (p. 707).

<table>
<thead>
<tr>
<th>Instances to collect information from</th>
<th>You can choose to inventory all instances in your AWS account, individually select instances, or target groups of instances by using Amazon EC2 tags. For more information about performing inventory collection on all of your instances, see</th>
</tr>
</thead>
</table>
Inventory

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inventory all managed instances in your AWS account (p. 719).</td>
</tr>
</tbody>
</table>

**When to collect information**

You can specify a collection interval in terms of minutes, hours, and days. The shortest collection interval is every 30 minutes.

**Note**

Depending on the amount of data collected, the system can take several minutes to report the data to the output you specified. After the information is collected, the metadata is sent over a secure HTTPS channel to a plain-text AWS store that is accessible only from your AWS account.

You can view the data in the AWS Systems Manager console on the **Inventory** page, which includes several predefined cards to help you query the data.
Learn more about Systems Manager Inventory

When you configure Systems Manager Inventory, you specify the type of metadata to collect, the instances from where the metadata should be collected, and a schedule for metadata collection. These configurations are saved with your AWS account as a Systems Manager State Manager association. An association is simply a configuration.

**Note**
Inventory only collects metadata. It does not collect any personal or proprietary data.

**Topics**
- Metadata collected by inventory (p. 707)
- Working with file and Windows registry inventory (p. 711)
- Related AWS services (p. 712)

**Metadata collected by inventory**

The following sample shows the complete list of metadata collected by each Inventory plugin.

```json
[
{
"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"}
]
}
{"name": "DHCPServer",  "dataType": "STRING"},
{"name": "DNSDataWay",  "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": "CompliantCount",  "dataType": "NUMBER"},
    {
      "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": "NonCompliantCount",  "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"
  ]
}
Note

With the release of version 2.5, RPM Package Manager replaced the Serial attribute with Epoch. The Epoch attribute is a monotonically increasing integer like Serial. When you inventory by
using the AWS:Application type, note that a larger value for Epoch means a newer version. If Epoch values are the same or empty, then use the value of the Version or Release attribute to determine the newer version.

Working with file and Windows registry inventory

Systems Manager Inventory enables you to search and inventory files on Windows and Linux operating systems. You can also search and inventory the Windows Registry.

Files: You can collect metadata information about files, including file names, the time files were created, the times files were last modified and accessed, and file sizes, to name a few. To start collecting file inventory, you specify a file path where you want to perform the inventory, one or more patterns that define the types of files you want to inventory, and if the path should be traversed recursively. Systems Manager inventories all file metadata for files in the specified path that match the pattern. File inventory uses the following parameter input.

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

- **Path**: The directory path where you want to inventory files. For Windows, you can use environment variables like `%PROGRAMFILES%` as long as the variable maps to a single directory path. For example, if you use `%PATH%` that maps to multiple directory paths, Inventory throws an error.
- **Pattern**: An array of patterns to identify files.
- **Recursive**: A Boolean value indicating whether Inventory should recursively traverse the directories.
- **DirScanLimit**: An optional value specifying how many directories to scan. Use this parameter to minimize performance impact on your instances. By default, Inventory scans a maximum of 5,000 directories.

**Note**

Inventory collects metadata for a maximum of 500 files across all specified paths.

Here are some examples of how to specify the parameters when performing an inventory of files.

- On Linux, collect metadata of .sh files in the `/home/ec2-user` directory, excluding all subdirectories.
  
  ```json
  [{
    "Path":"/home/ec2-user",
    "Pattern": [ "*.sh",  "*.sh" ],
    "Recursive":false
  }]
  ```

- On Windows, collect metadata of all `.exe` files in the Program Files folder, including subdirectories recursively.
  
  ```json
  [{
    "Path":"C:\Program Files",
    "Pattern": [ "*.exe" ],
    "Recursive":true
  }]
  ```

- On Windows, collect metadata of specific log patterns.
  
  ```json
  [{
    "Path":"C:\ProgramData\Amazon",
    "Pattern": [ "amazon*.log" ],
    "Recursive":true
  }]
  ```

- Limit the directory count when performing recursive collection.
  
  ```json
  [{
    "Path":"C:\Users",
    "Pattern": [ "*.ps1" ],
    "Recursive":true,
    "DirScanLimit": 1000
  }]
  ```
**Windows Registry:** You can collect Windows Registry keys and values. You can choose a key path and collect all keys and values recursively. You can also collect a specific registry key and its value for a specific path. Inventory collects the key path, name, type, and the value.

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

- **Path:** The path to the Registry key.
- **Recursive:** A Boolean value indicating whether Inventory should recursively traverse Registry paths.
- **ValueNames:** An array of value names for performing inventory of Registry keys. If you use this parameter, Systems Manager will inventory only the specified value names for the specified path.

**Note**
Inventory collects a maximum of 250 Registry key values for all specified paths.

Here are some examples of how to specify the parameters when performing an inventory of the Windows Registry.

- Collect all keys and values recursively for a specific path.

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

- Collect all keys and values for a specific path (recursive search disabled).

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

- Collect a specific key by using the ValueNames option.

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

**Related AWS services**
Systems Manager Inventory provides a snapshot of your current inventory to help you manage software policy and improve the security posture of your entire fleet. You can extend your inventory management and migration capabilities using the following AWS services.

- **AWS Config** provides a historical record of changes to your inventory, along with the ability to create rules to generate notifications when a configuration item is changed. For more information, see, Recording Amazon EC2 managed instance inventory in the *AWS Config Developer Guide*.
- **AWS Application Discovery Service** is designed to collect inventory on OS type, application inventory, processes, connections, and server performance metrics from your on-premises VMs to support a successful migration to AWS. For more information, see the *Application Discovery Service User Guide*.

**Configuring Resource Data Sync for Inventory**
This topic describes how to set up and configure resource data sync for Systems Manager Inventory. For information about resource data sync for Systems Manager Explorer, see *Setting up Systems Manager Explorer to display data from multiple accounts and Regions* (p. 127).
## About Resource Data Sync

You can use Systems Manager resource data sync to send inventory data collected from all of your managed instances to a single S3 bucket. Resource data sync then automatically updates the centralized data when new inventory data is collected. With all inventory data stored in a target S3 bucket, you can use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data.

For example, say that you've configured inventory to collect data about the operating system (OS) and applications running on a fleet of 150 managed instances. Some of these instances are located in a hybrid data center, and others are running in Amazon EC2 across multiple AWS Regions. If you have not configured resource data sync, you either need to manually gather the collected inventory data for each instance, or you have to create scripts to gather this information. You would then need to port the data into an application so that you can run queries and analyze it.

With resource data sync, you perform a one-time operation that synchronizes all inventory data from all of your managed instances. After the sync is successfully created, Systems Manager creates a baseline of all inventory data and saves it in the target S3 bucket. When new inventory data is collected, Systems Manager automatically updates the data in the S3 bucket. You can then quickly and cost-effectively port the data to Amazon Athena and Amazon QuickSight.

Diagram 1 shows how resource data sync aggregates inventory data from managed instances in Amazon EC2 and a hybrid environment to a target S3 bucket. This diagram also shows how resource data sync works with multiple AWS accounts and AWS Regions.

**Diagram 1: Resource Data Sync with Multiple AWS Accounts and AWS Regions**

If you delete a managed instance, resource data sync preserves the inventory file for the deleted instance. For running instances, however, resource data sync automatically overwrites old inventory files when new files are created and written to the S3 bucket. If you want to track inventory changes over time, you can use the AWS Config service to track the SSM::ManagedInstanceInventory resource type. For more information, see [Getting Started with AWS Config](#).

Use the procedures in this section to create a resource data sync for Inventory by using the Amazon S3 and AWS Systems Manager consoles. You can also use AWS CloudFormation to create or delete a resource data sync. To use AWS CloudFormation, add the `AWS::SSM::ResourceDataSync` resource to your AWS CloudFormation template. For information, see one of the following documentation resources:
• AWS CloudFormation Resource for Resource Data Sync in AWS Systems Manager (blog)
• Working with AWS CloudFormation Templates in the AWS CloudFormation User Guide

Note
You can use AWS Key Management Service (AWS KMS) to encrypt inventory data in the S3 bucket. For an example of how to create an encrypted sync by using the AWS CLI and how to work with the centralized data in Amazon Athena and Amazon QuickSight, see Walkthrough: Use Resource Data Sync to aggregate inventory data (p. 751).

Before you begin
Before you create a resource data sync, use the following procedure to create a central S3 bucket to store aggregated inventory data. The procedure describes how to assign a bucket policy that enables Systems Manager to write inventory data to the bucket from multiple accounts. If you already have an S3 bucket that you want to use to aggregate inventory data for resource data sync, then you must configure the bucket to use the policy in the following procedure.

To create and configure an S3 bucket for resource data sync
1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket to store your aggregated Inventory data. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. Choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace bucket-name and account-id with the name of the S3 bucket you created and a valid AWS account ID.

To enable multiple AWS accounts to send inventory data to the central Amazon S3 bucket, specify each account in the policy as shown in the following Resource sample:

```
"Resource": [
    "arn:aws:s3:::MyTestS3Bucket/*accountid=123456789012/*/",
    "arn:aws:s3:::MyTestS3Bucket/*accountid=a1b2c3d4e5f6/*/",
    "arn:aws:s3:::MyTestS3Bucket/*accountid=1234abcd56ef/*/"
],
```

Optionally, replace bucket-prefix with the name of an Amazon S3 prefix (subdirectory). If you didn't create a prefix, remove bucket-prefix/ from the ARN in the following policy.

Note
For information about viewing your AWS account ID, see Your AWS Account ID and Its Alias in the IAM User Guide.
Use the following procedure to create a resource data sync for Systems Manager Inventory by using the Systems Manager console. For information about how to create a resource data sync by using the AWS CLI, see Walkthrough: Configure your managed instances for Inventory by using the CLI (p. 749).

**To create a Resource Data Sync**


2. In the navigation pane, choose **Managed Instances**.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Managed Instances**.

3. Choose **Configure Inventory, Resource Data Syncs**, and then choose **Create resource data sync**.

4. In the **Sync name** field, type a name for the sync configuration.

5. In the **Bucket name** field, type the name of the Amazon S3 bucket you created at the start of this procedure.

6. (Optional) In the **Bucket prefix** field, type the name of an S3 bucket prefix (subdirectory).

7. In the **Bucket region** field, choose **This region** if the S3 bucket you created is located in the current AWS Region. If the bucket is located in a different AWS Region, choose **Another region**, and type the name of the Region.

---

**Note**

The Asia Pacific Region came online in April 25, 2019. If you create a resource data sync for an AWS Region that came online since the Asia Pacific (Hong Kong) Region (ap-east-1) or later, then you must enter a region-specific service principal entry in the **SSMBucketDelivery** section. The following example includes a region-specific service principal entry for ssm.ap-east-1.amazonaws.com.

```json
{
    "Sid": "SSMBucketDelivery",
    "Effect": "Allow",
    "Principal": {
        "Service": ["ssm.amazonaws.com", "ssm.ap-east-1.amazonaws.com"]
    }
}
```
8. (Optional) In the **KMS Key ARN** field, type or paste a KMS Key ARN to encrypt inventory data in Amazon S3.

9. Choose **Create**.

To synchronize inventory data from multiple AWS Regions, you must create a resource data sync in each Region. Repeat this procedure in each AWS Region where you want to collect inventory data and send it to the central S3 bucket. When you create the sync in each Region, specify the central Amazon S3 bucket in the **Bucket name** field. Then use the **Bucket region** option to choose the Region where you created the central Amazon S3 bucket, as shown in the following screen shot. The next time the association runs to collect inventory data, Systems Manager stores the data in the central S3 bucket.

### Creating an Inventory Resource Data Sync for multiple accounts defined in AWS Organizations

You can synchronize inventory data from multiple AWS accounts to a central Amazon S3 bucket. The accounts must be defined in AWS Organizations. After you complete the following procedure, inventory data is synchronized to individual Amazon S3 key prefixes in the central bucket. Each key prefix represents a different AWS account ID.

**Note**
Before you begin, verify that you set up and configured multiple AWS accounts in AWS Organizations. For more information, see in the [AWS Organizations User Guide](https://docs.aws.amazon.com/organizations/latest/userguide/orgs_user_guide.html).

**Before you begin**

Before you create a resource data sync for multiple accounts defined in AWS Organizations, use the following procedure to create a central S3 bucket to store aggregated inventory data. The procedure describes how to assign a bucket policy that enables Systems Manager to write inventory data to the bucket from your AWS Organizations account ID. If you already have an S3 bucket that you want to use to aggregate inventory data for Resource Data Sync, then you must configure the bucket to use the policy in the following procedure.
To create and configure an S3 bucket for resource data sync for multiple accounts defined in AWS Organizations

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket to store your aggregated inventory data. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. Choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace `bucket-name` and `organization-id` with the name of the Amazon S3 bucket you created and a valid AWS Organizations account ID.

   Optionally, replace `bucket-prefix` with the name of an Amazon S3 prefix (subdirectory). If you didn't create a prefix, remove `bucket-prefix/` from the ARN in the following policy.

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

   **Note**
   The Asia Pacific Region came online in April 25, 2019. If you create a resource data sync for an AWS Region that came online since the Asia Pacific (Hong Kong) Region
When you configure inventory collection, you start by creating a Systems Manager State Manager association. Systems Manager collects the inventory data when the association is run. If you don’t create the association first, and attempt to invoke the `aws:softwareInventory` plugin by using, for example, Run Command, the system returns the following error:

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

Also note that an instance can have only one inventory association configured at a time. If you configure an instance with two or more associations, the inventory association doesn’t run and no inventory data is collected.

**Before You Begin**

Before you configure inventory collection, complete the following tasks.

- Update SSM Agent on the instances you want to inventory. By running the latest version of SSM Agent, you ensure that you can collect metadata for all supported inventory types. For information
Inventory all managed instances in your AWS account

You can easily inventory all managed instances in your AWS account by creating a global inventory association. A global inventory association performs the following actions:

- Automatically applies the global inventory configuration (association) to all existing managed instances in your AWS account. Instances that already have an inventory association are skipped when the global inventory association is applied and runs. When an instance is skipped, the detailed status message states **Overridden By Explicit Inventory Association**. Those instances are skipped by the global association, but they will still report inventory when they run their assigned inventory association.

- Automatically adds new instances created in your AWS account to the global inventory association.

**Note**

- If an instance is configured for the global inventory association, and you assign a specific association to that instance, then Systems Manager Inventory deprioritizes the global association and applies the specific association.

- Global inventory associations are available in SSM Agent version 2.0.790.0 or later. For information about how to update SSM Agent on your instances, see **Update SSM Agent by using Run Command (p. 835)**.

Configuring inventory collection with one click (console)

Use the following procedure to configure Systems Manager Inventory for all managed instances in your AWS account and in a single AWS Region.

**To configure all of your managed instances in the current Region for Systems Manager inventory**

2. In the navigation pane, choose **Inventory**.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Inventory** in the navigation pane.
3. In the **Managed instances with inventory enabled** card, choose **Click here to enable inventory on all instances**.
Configuring inventory collection

If successful, the console displays the following message.

Depending on the number of managed instances in your account, it can take several minutes for the global inventory association to be applied. Wait a few minutes and then refresh the page. Verify that the graphic changes to reflect that inventory is configured on all of your managed instances.

Configuring collection by using the console

This section includes information about how to configure Systems Manager Inventory to collect metadata from your managed instances by using the Systems Manager console. You can quickly collect metadata from all instances in a specific AWS account (and any future instances that might be created in that account) or you can selectively collect inventory data by using tags or instance IDs.

To configure inventory collection

2. In the navigation pane, choose Inventory.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Inventory in the navigation pane.
3. Choose Setup Inventory.
4. In the Targets section, identify the instances where you want to run this operation by choosing one of the following.
   
   • **Selecting all managed instances in this account** - This option selects all managed instances for which there is no existing inventory association. If you choose this option, instances that already had inventory associations are skipped during inventory collection, and shown with a status of Skipped in inventory results. For more information, see Inventory all managed instances in your AWS account (p. 719).
   
   • **Specifying a tag** - This option lets you specify a single tag to identify instances in your account from which you want to collect inventory. If you use a tag, any instances created in the future with the same tag will also report inventory. If there is an existing inventory association with all instances, using a tag to select specific instances as a target for a different inventory overrides instance membership in the All managed instances target group. Instances with the specified tag are skipped on future inventory collection from All managed instances.
   
   • **Manually selecting instances** - This option lets you choose specific managed instances in your account. Explicitly choosing specific instances by using this option overrides inventory associations
on the All managed instances target. The instance is skipped on future inventory collection from All managed instances.

5. In the Schedule section, choose how often you want the system to collect inventory metadata from your instances.

6. In the Parameters section, use the lists to enable or disable different types of inventory collection. See the following samples if you want to create an inventory search for Files or the Windows Registry.

**Files**

- On Linux, collect metadata of .sh files in the /home/ec2-user directory, excluding all subdirectories.

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

- On Windows, collect metadata of all *.exe files in the Program Files folder, including subdirectories recursively.

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

- On Windows, collect metadata of specific log patterns.

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

- Limit the directory count when performing recursive collection.

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

**Windows registry**

- Collect all keys and values recursively for a specific path.

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

- Collect all keys and values for a specific path (recursive search disabled).

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

- Collect a specific key by using the ValueNames option.

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

For more information about collecting File and Windows Registry inventory, see Working with file and Windows registry inventory (p. 711).

7. In the Advanced section, choose Sync inventory execution logs to an S3 bucket if you want to store the association execution status in an S3 bucket.

8. Choose Setup Inventory. Systems Manager creates a State Manager association and immediately runs Inventory on the instances.

9. In the navigation pane, choose State Manager. Verify that a new association was created that uses the AWS-GatherSoftwareInventory document. The association schedule uses a rate expression. Also, verify that the Status field shows Success. If you chose the option to Sync inventory execution logs to an S3 bucket, then you can view the log data in Amazon S3 after a few minutes. If you want
to view inventory data for a specific instance, then choose Managed Instances in the navigation pane.

10. Choose an instance, and then choose View details.
11. On the instance details page, choose Inventory. Use the Inventory type lists to filter the inventory.

### Working with Systems Manager inventory data

This section includes topics that describe how to query and aggregate inventory data.

**Topics**
- Querying inventory data from multiple Regions and accounts (p. 722)
- Querying an inventory collection by using filters (p. 727)
- Aggregating inventory data (p. 727)

### Querying inventory data from multiple Regions and accounts

AWS Systems Manager Inventory integrates with Amazon Athena to help you query inventory data from multiple AWS Regions and accounts. Athena integration uses resource data sync so that you can view inventory data from all of your managed instances on the **Inventory Detail View** page in the AWS Systems Manager console.

**Important**

This feature uses AWS Glue to crawl the data in your Amazon Simple Storage Service (Amazon S3) bucket, and Amazon Athena to query the data. Depending on how much data is crawled and queried, you can be charged for using these services. With AWS Glue, you pay an hourly rate, billed by the second, for crawlers (discovering data) and ETL jobs (processing and loading data). With Athena, you are charged based on the amount of data scanned by each query. We encourage you to view the pricing guidelines for these services before you use Amazon Athena integration with Systems Manager Inventory. For more information, see Amazon Athena pricing and AWS Glue pricing.

You can view inventory data on the **Inventory Detail View** page in all AWS Regions where Amazon Athena is available. For a list of supported Regions, see Amazon Athena Service Endpoints in the Amazon Web Services General Reference.

**Before you begin**

Athena integration uses resource data sync. You must set up and configure resource data sync to use this feature. For more information, see Configuring Resource Data Sync for Inventory (p. 712).

Also, be aware that the **Inventory Detail View** page displays inventory data for the owner of the central S3 bucket used by resource data sync. If you are not the owner of the central S3 bucket, then you won't see inventory data on the **Inventory Detail View** page.

### Configuring access

Before you can query and view data from multiple accounts and Regions on the **Inventory Detail View** page in the Systems Manager console, you must configure your AWS Identity and Access Management (IAM) user account. The following procedure describes how to use the IAM console to configure your IAM user account so that you can view inventory data on the **Inventory Detail View** page.

**To configure access to the inventory detail view page**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users, and then choose the user account you want to configure. The Summary page opens.
3. On the Permissions tab, choose Add permissions.
4. On the Grant permissions page, choose Attach existing policies directly.
5. In the Search field, search for AWSQuicksightAthenaAccess.
6. Choose the option next to this policy, and then choose Next: Review.
7. Choose Add permissions.
8. Choose the user name again to return to the Summary page.
9. Now add an inline policy so that AWS Glue can crawl your inventory data. On the Permissions tab, at the right side of the page, choose Add inline policy. The Create policy page opens.
10. Choose the JSON tab.
11. Delete the existing JSON text in the editor, and then copy and paste the following policy into the JSON editor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetCrawlers",
        "glue:GetCrawler",
        "glue:GetTables",
        "glue:StartCrawler",
        "glue:CreateCrawler"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole",
        "iam:CreateRole",
        "iam:AttachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:account_ID:role/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreatePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:account_ID:policy/*"
      ]
    }
  ]
}
```

**Note**  
(Optional) If the Amazon Simple Storage Service (Amazon S3) bucket used to store inventory data is encrypted by using the AWS Key Management Service, then you must also add the following block to the policy.

```
{
  "Effect": "Allow",
  "Action": [
    "kms:GrantWriteSession",
    "kms:Encrypt"
  ],
  "Resource": [
    "arn:aws:kms::*:key/*"
  ]
}
If you paste this block after the last block in the policy, be sure to separate the blocks with a comma (,).

12. On the Review Policy page, enter a name in the Name field.
13. Choose Create policy.

Important
When you choose a resource data sync on the Inventory Detail View page, Systems Manager automatically creates the Amazon-GlueServiceRoleForSSM role. This role enables AWS Glue to access the S3 bucket for resource data sync. Systems Manager automatically attaches the following policies to the role:

- Amazon-GlueServicePolicyForSSM-{Amazon S3 bucket name}: This policy enables communication between AWS Glue and Systems Manager Inventory.
- AWSGlueServiceRole: This is an AWS managed policy that enables access to AWS Glue.

If a policy with the name Amazon-GlueServicePolicyForSSM-{S3 bucket name} already exists in your IAM user account, and this policy is not attached to the Amazon-GlueServiceRoleForSSM role, then the system returns an error. To resolve this issue, use the IAM console to verify that the contents of the Amazon-GlueServicePolicyForSSM-{Amazon S3 bucket name} policy match the inline policy in this procedure. Then attach the policy to the Amazon-GlueServiceRoleForSSM role.

Querying data on the inventory detailed view page

Use the following procedure to view inventory data from multiple AWS Regions and accounts on the Systems Manager Inventory Detailed View page.

Important
The Inventory Detailed View page is only available in AWS Regions that offer Amazon Athena. If the following tabs are not displayed on the Systems Manager Inventory page, it means Athena is not available in the Region and you can't use the Detailed View to query data.

To view inventory data from multiple Regions and accounts in the AWS Systems Manager console

2. In the navigation pane, choose Inventory.

-or-
If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Inventory** in the navigation pane.

3. Choose the **Detailed View** tab.

4. Choose the resource data sync for which you want to query data.

5. In the **Inventory Type** list, choose the type of inventory data that you want to query, and then press **Enter**.

6. To filter the data, choose the Filter bar, and then choose a filter option.

The following example shows AWSComponent inventory data filtered on the us-east-2 Region.
You can use the **Export to CSV** button to view the current query set in a spreadsheet application such as Microsoft Excel. You can also use the **Query History** and **Run Advanced Queries** buttons to view history details and interact with your data in Amazon Athena.

**Editing the AWS Glue crawler schedule**

AWS Glue crawls the inventory data in the central S3 bucket twice daily, by default. If you frequently change the types of data to collect on your instances then you might want to crawl the data more frequently, as described in the following procedure.

**Important**

AWS Glue charges your account based on an hourly rate, billed by the second, for crawlers (discovering data) and ETL jobs (processing and loading data). Before you change the crawler schedule, view the AWS Glue pricing page.

**To change the inventory data crawler schedule**

2. In the navigation pane, choose **Crawlers**.
3. In the crawlers list, choose the option next to the Systems Manager Inventory data crawler. The crawler name uses the following format:

   `AWSSystemsManager-Resource_Data_Sync_bucket_name-Region-AWS_account_ID`

4. Choose **Action**, and then choose **Edit crawler**.
5. In the navigation pane, choose **Schedule**.
6. In the **Cron expression** field, specify a new schedule by using a cron format. For more information about the cron format, see *Time-Based Schedules for Jobs and Crawlers* in the *AWS Glue Developer Guide*.
Important
You can pause the crawler to stop incurring charges from AWS Glue. If you pause the crawler, or if you change the frequency so that the data is crawled less often, then the Inventory Detailed View might display data that is not current.

Querying an inventory collection by using filters

After you collect inventory data, you can use the filter capabilities in Systems Manager to query a list of managed instances that meet certain filter criteria.

To query instances based on inventory filters
2. In the navigation pane, choose Inventory.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Inventory in the navigation pane.
3. In the Filter by resource groups, tags or inventory types section, choose the filter box. A list of predefined filters appears.
4. Choose an attribute to filter on. For example, choose AWS:Application. If prompted, choose a secondary attribute to filter. For example, choose AWS:Application.Name.
5. Choose a delimiter from the list. For example, choose Begin with. A text box appears in the filter.
6. Type a value in the text box. For example, type Amazon (SSM Agent is named Amazon SSM Agent).
7. Press Enter. The system returns a list of managed instances that include an application name that begins with the word Amazon.

Note
You can combine multiple filters to refine your search.

Aggregating inventory data

After you configure your managed instances for AWS Systems Manager Inventory, you can view aggregated counts of inventory data. For example, say you configured dozens or hundreds of managed instances to collect the AWS:Application inventory type. By using the information in this section, you can see an exact count of how many instances are configured to collect this data.

You can also see specific inventory details by aggregating on a data type. For example, the AWS:InstanceInformation inventory type collects operating system platform information with the Platform data type. By aggregating data on the Platform data type, you can quickly see how many instances are running Windows and how many are running Linux.

The procedures in this section describe how to view aggregated counts of inventory data by using the AWS CLI. You can also view pre-configured aggregated counts in the AWS Systems Manager console on the Inventory page. These pre-configured dashboards are called Inventory Insights and they offer one-click remediation of your inventory configuration issues.

Note the following important details about aggregation counts of inventory data:

- Systems Manager Inventory stores inventory data for 30 days. This means that aggregated counts of inventory include all data collected during the last 30 days.
- Inventory shows data that has been sent by an instance over the course of its lifetime. If an instance was previously configured to report a specific inventory data type, for example AWS:Network, and later you change the configuration to stop collecting that type, aggregation counts still show AWS:Network data until the instance has been terminated.
• If an instance was previously configured to collect inventory data, and you terminate that instance, inventory counts still show data for the deleted instance for 30 days.

For information about how to quickly configure and collect inventory data from all instances in a specific AWS account (and any future instances that might be created in that account) see Configuring collection by using the console (p. 720).

Topics
• Aggregating inventory data to see counts of instances that collect specific types of data (p. 728)
• Aggregating inventory data with groups to see which instances are and aren’t configured to collect an inventory type (p. 732)

Aggregating inventory data to see counts of instances that collect specific types of data

You can use the GetInventory API action to view aggregated counts of instances that collect one or more inventory types and data types. For example, the AWS:InstanceInformation inventory type enables you to view an aggregate of operating systems by using the GetInventory API action with the AWS:InstanceInformation.PlatformType data type. Here is an example AWS CLI command and output:

```bash
aws ssm get-inventory --aggregators "Expression=AWS:InstanceInformation.PlatformType"
```

The system returns information like the following.

```json
{
   "Entities":[
       {
           "Data":{
               "AWS:InstanceInformation":{
                   "Content":[
                       {
                           "Count":7,
                           "PlatformType":"windows"
                       },
                       {
                           "Count":5,
                           "PlatformType":"linux"
                       }
                   ]
               }
           }
       }
   ]
}
```

Getting started

Determine the inventory types and data types for which you want to view counts. You can view a list of inventory types and data types that support aggregation by running the following command in the AWS CLI:

```bash
aws ssm get-inventory-schema --aggregator
```

The command returns a JSON list of inventory types and data types that support aggregation. The TypeName field shows supported inventory types. And the Name field shows each data type. For
example, in the following list, the AWS:Application inventory type includes data types for Name and Version.

```
{
    "Schemas": [
    {
        "TypeName": "AWS:Application",
        "Version": "1.1",
        "DisplayName": "Application",
        "Attributes": [
            {
                "DataType": "STRING",
                "Name": "Name"
            },
            {
                "DataType": "STRING",
                "Name": "Version"
            }
        ]
    },
    {
        "TypeName": "AWS:InstanceInformation",
        "Version": "1.0",
        "DisplayName": "Platform",
        "Attributes": [
            {
                "DataType": "STRING",
                "Name": "PlatformName"
            },
            {
                "DataType": "STRING",
                "Name": "PlatformType"
            },
            {
                "DataType": "STRING",
                "Name": "PlatformVersion"
            }
        ]
    },
    {
        "TypeName": "AWS:ResourceGroup",
        "Version": "1.0",
        "DisplayName": "ResourceGroup",
        "Attributes": [
            {
                "DataType": "STRING",
                "Name": "Name"
            }
        ]
    },
    {
        "TypeName": "AWS:Service",
        "Version": "1.0",
        "DisplayName": "Service",
        "Attributes": [
            {
                "DataType": "STRING",
                "Name": "Name"
            },
            {
                "DataType": "STRING",
                "Name": "DisplayName"
            },
            {
                "DataType": "STRING",
                "Name": "ContextResource"
            }
        ]
    }
}
```
You can aggregate data for any of the listed inventory types by creating a command that uses the following syntax:

```bash
aws ssm get-inventory --aggregators "Expression=InventoryType.DataType"
```

Here are some examples.

**Example 1**

This example aggregates a count of the Windows roles used by your instances.

```bash
aws ssm get-inventory --aggregators "Expression=AWS:WindowsRole.Name"
```

**Example 2**

This example aggregates a count of the applications installed on your instances.

```bash
aws ssm get-inventory --aggregators "Expression=AWS:Application.Name"
```

**Combining Multiple Aggregators**

You can also combine multiple inventory types and data types in one command to help you better understand the data. Here are some examples.
Example 1
This example aggregates a count of the operating system types used by your instances. It also returns the specific name of the operating systems.

```bash
aws ssm get-inventory --aggregators '[["Expression": "AWS:InstanceInformation.PlatformType", "Aggregators": [["Expression": "AWS:InstanceInformation.PlatformName"]]]]
```

Example 2
This example aggregates a count of the applications running on your instances and the specific version of each application.

```bash
```

If you prefer, you can create an aggregation expression with one or more inventory types and data types in a JSON file and call the file from the AWS CLI. The JSON in the file must use the following syntax:

```json
[
  {
    "Expression": "string",
    "Aggregators": [
      {
        "Expression": "string"
      }
    ]
  }
]
```

You must save the file with the .json file extension.

Here is an example that uses multiple inventory types and data types.

```json
[
  {
    "Expression": "AWS:Application.Name",
    "Aggregators": [
      {
        "Expression": "AWS:Application.Version",
        "Aggregators": [
          {
            "Expression": "AWS:InstanceInformation.PlatformType"
          }
        ]
      }
    ]
  }
]
```

Use the following command to call the file from the AWS CLI.

```bash
aws ssm get-inventory --aggregators file://file_name.json
```

The command returns information like the following:

```json
{"Entities": [
```
Aggregating inventory data with groups to see which instances are and aren't configured to collect an inventory type

Groups enable you to quickly see a count of which managed instances are and aren't configured to collect one or more inventory types. With groups, you specify one or more inventory types and a filter that uses the `exists` operator.

For example, say that you have four managed instances configured to collect the following inventory types:

- Instance 1: AWS:Application
- Instance 2: AWS:File
- Instance 3: AWS:Application, AWS:File
- Instance 4: AWS:Network

You can run the following command from the AWS CLI to see how many instances are configured to collect both the AWS:Application and AWS:File inventory types. The response also returns a count of how many instances aren't configured to collect both of these inventory types.

```bash
aws ssm get-inventory -- aggregators
'Groups=[{Name=ApplicationAndFile,Filters=([{Key=TypeName,Values=[AWS:Application],Type=Exists},
{Key=TypeName,Values=[AWS:File],Type=Exists})]}]
```

The command response shows that only one managed instance is configured to collect both the AWS:Application and AWS:File inventory types.
Note
Groups don't return data type counts. Also, you can't drill-down into the results to see the IDs of instances that are or aren't configured to collect the inventory type.

If you prefer, you can create an aggregation expression with one or more inventory types in a JSON file and call the file from the AWS CLI. The JSON in the file must use the following syntax:

```json
{
    "Aggregators": [
        {
            "Groups": [
                {
                    "Name": "Name",
                    "Filters": [
                        {
                            "Key": "TypeName",
                            "Values": [
                                "Inventory_type"
                            ],
                            "Type": "Exists"
                        },
                        {
                            "Key": "TypeName",
                            "Values": [
                                "Inventory_type"
                            ],
                            "Type": "Exists"
                        }
                    ]
                }
            ]
        }
    ]
}
```

You must save the file with the .json file extension.

Use the following command to call the file from the AWS CLI.

```
aws ssm get-inventory --cli-input-json file://file_name.json
```

Additional examples
The following examples show you how to aggregate inventory data to see which managed instances are and aren't configured to collect the specified inventory types. These examples use the AWS CLI. Each example includes a full command with filters that you can run from the command line and a sample input.json file if you prefer to enter the information in a file.

**Example 1**

This example aggregates a count of instances that are and aren't configured to collect either the AWS:Application or the AWS:File inventory types.

Run the following command from the AWS CLI.

```bash
aws ssm get-inventory --aggregators
'Groups=[{Name=ApplicationORFile,Filters=[{Key=TypeName,Values=[AWS:Application,
AWS:File],Type=Exists}]}]' 
```

If you prefer to use a file, copy and paste the following sample into a file and save it as input.json.

```json
{
   "Aggregators": [
      {
         "Groups": [
            {
               "Name": "ApplicationORFile",
               "Filters": [
                  {
                     "Key": "TypeName",
                     "Values": [
                        "AWS:Application",
                        "AWS:File"
                     ],
                     "Type": "Exists"
                  }
               ]
            }
         ]
      }
   ]
}
```

Run the following command from the AWS CLI.

```bash
aws ssm get-inventory --cli-input-json file://input.json 
```

The command returns information like the following:

```json
{
   "Entities": [
      {
         "Data": {
            "ApplicationORFile": {
               "Content": [
                  {
                     "notMatchingCount": "1"
                  },
                  {
                     "matchingCount": "3"
                  }
               ]
            }
         }
      }
   ]
}
```
Example 2

This example aggregates a count of instances that are and aren't configured to collect the AWS:Application, AWS:File, and AWS:Network inventory types.

Run the following command from the AWS CLI.

```bash
aws ssm get-inventory --aggregators
  'Groups=[[Name=Application,Filters=[{Key=TypeName,Values=[AWS:Application],Type=Exists}]],
  {Name=File,Filters=[{Key=TypeName,Values=[AWS:File],Type=Exists}]],
  {Name=Network,Filters=[{Key=TypeName,Values=[AWS:Network],Type=Exists}]}]
```

If you prefer to use a file, copy and paste the following sample into a file and save it as input.json.

```json
{
  "Aggregators": [
    {
      "Groups": [ 
        {
          "Name": "Application",
          "Filters": [ 
            {
              "Key": "TypeName",
              "Values": [ 
                "AWS:Application"
              ],
              "Type": "Exists"
            }
          ]
        },
        {
          "Name": "File",
          "Filters": [ 
            {
              "Key": "TypeName",
              "Values": [ 
                "AWS:File"
              ],
              "Type": "Exists"
            }
          ]
        },
        {
          "Name": "Network",
          "Filters": [ 
            {
              "Key": "TypeName",
              "Values": [ 
                "AWS:Network"
              ],
              "Type": "Exists"
            }
          ]
        }
      ]
    }
  ]
}
```
Run the following command from the AWS CLI.

```
aws ssm get-inventory --cli-input-json file://input.json
```

The command returns information like the following:

```
{
  "Entities": [
    {
      "Data": {
        "Application": {
          "Content": [
            {
              "notMatchingCount": "2"
            },
            {
              "matchingCount": "2"
            }
          ]
        },
        "File": {
          "Content": [
            {
              "notMatchingCount": "2"
            },
            {
              "matchingCount": "2"
            }
          ]
        },
        "Network": {
          "Content": [
            {
              "notMatchingCount": "3"
            },
            {
              "matchingCount": "1"
            }
          ]
        }
      }
    }
  ]
}
```

**Working with custom inventory**

You can assign any metadata you want to your instances by creating custom inventory. For example, let's say you manage a large number of servers in racks in your data center, and these servers have been configured as Systems Manager managed instances. Currently, you store information about server rack location in a spreadsheet. With custom inventory, you can specify the rack location of each instance as metadata on the instance. When you collect inventory by using Systems Manager, the metadata is collected with other inventory metadata. You can then port all inventory metadata to a central Amazon S3 bucket by using Resource Data Sync and query the data.

**Note**

Systems Manager supports a maximum of 20 custom inventory types per AWS account.

To assign custom inventory to an instance, you can either use the Systems Manager `PutInventory` API action, as described in *Walkthrough: Assign custom inventory metadata to an instance* (p. 748). Or,
you can create a custom inventory JSON file and upload it to the instance. This section describes how to create the JSON file.

The following example JSON file with custom inventory specifies rack information about an on-premises server. This example specifies one type of custom inventory data ("TypeName": "Custom:RackInformation"), with multiple entries under Content that describe the data.

```json
{
    "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"
    }
}
```

You can also specify distinct entries in the Content section, as shown in the following example.

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

The JSON schema for custom inventory requires SchemaVersion, TypeName, and Content sections, but you can define the information in those sections.

```json
{
    "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 is limited to 100 characters. Also, the TypeName section must start with Custom. For example, Custom:PuppetModuleInfo. Both Custom and the Data you specify must begin with a capital letter. The following examples would cause an exception: "CUSTOM:RackInformation", "custom.rackinformation".

The Content section includes attributes and data. These items are not case-sensitive. However, if you define an attribute (for example: "Vendor": "DELL"), then you must consistently reference this attribute in your custom inventory files. If you specify "Vendor": "DELL" (using a capital “V” in vendor) in one file, and then you specify "vendor": "DELL" (using a lowercase “v” in vendor) in another file, the system returns an error.
You must save the file with a .json extension and the inventory you define must consist only of string values.

After you create the file, you must save it on the instance. The following table shows the location where custom inventory JSON files must be stored on the instance:

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>%SystemDrive%\ProgramData\Amazon\SSM \InstanceData&lt;instance-id&gt;\inventory\custom</td>
</tr>
<tr>
<td>Linux</td>
<td>/var/lib/amazon/ssm/&lt;instance-id&gt;/inventory/custom</td>
</tr>
</tbody>
</table>

For an example of how to use custom inventory, see Get Disk Utilization of Your Fleet Using EC2 Systems Manager Custom Inventory Types.

Deleting custom inventory

You can use the **DeleteInventory** API action to delete a custom inventory type and the data associated with that type. You call the delete-inventory command by using the AWS CLI to delete all data for an inventory type. You call the delete-inventory command with the **SchemaDeleteOption** to delete a custom inventory type.

Note
An inventory type is also called an inventory schema.

The **SchemaDeleteOption** parameter includes the following options:

- **DeleteSchema**: This option deletes the specified custom type and all data associated with it. You can recreate the schema later, if you want.
- **DisableSchema**: If you choose this option, the system disables the current version, deletes all data for it, and ignores all new data if the version is less than or equal to the disabled version. You can enable this inventory type again by calling the **PutInventory** action for a version greater than the disabled version.

To delete or disable custom inventory by using the AWS CLI

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to use the **dry-run** option to see which data will be deleted from the system. This command doesn't delete any data.

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

   The system returns information like the following.

   ```json
   {
     "DeletionSummary":{
       "RemainingCount":3,
       "SummaryItems":[
         {
           "Count":2,
         }
     }
   }
   ```
For information about how to understand the delete inventory summary, see Understanding the delete inventory summary (p. 742).

#### 3. Run the following command to delete all data for a custom inventory type.

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

**Note**
The output of this command doesn't show the deletion progress. For this reason, TotalCount and Remaining Count are always the same because the system has not deleted anything yet. You can use the describe-inventory-deletions command to show the deletion progress, as described later in this topic.

The system returns information like the following.

```json
{
    "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"
}
```

The system deletes all data for the specified custom inventory type from the Systems Manager Inventory service.

#### 4. Run the following command. The command performs the following actions for the current version of the inventory type: disables the current version, deletes all data for it, and ignores all new data if the version is less than or equal to the disabled version.

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

The system returns information like the following.
You can view a disabled inventory type by using the following command.

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

5. Run the following command to delete an inventory type.

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

The system deletes the schema and all inventory data for the specified custom type.

The system returns information like the following.

```json
{
  "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"
}
```
Viewing the deletion status

You can check the status of a delete operation by using the describe-inventory-deletions AWS CLI command. You can specify a deletion ID to view the status of a specific delete operation. Or, you can omit the deletion ID to view a list of all deletions run in the last 30 days.

1. Run the following command to view the status of a deletion operation. The system returned the deletion ID in the delete-inventory summary.

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

   The system returns the latest status. The delete operation might not be finished yet. The system returns information like the following.

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

   If the delete operation is successful, the LastStatusMessage states: Deletion is successful.

   ```json
   {"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. Run the following command to view a list of all deletions run in the last 30 days.

   ```bash
   aws ssm describe-inventory-deletions --max-results a number
   ```
Understanding the delete inventory summary

To help you understand the contents of the delete inventory summary, consider the following example. A user assigned Custom:RackSpace inventory to three instances. Inventory items 1 and 2 use custom type version 1.0 ("SchemaVersion": "1.0"). Inventory item 3 uses custom type version 2.0 ("SchemaVersion": "2.0").

RackSpace custom inventory 1

```json
{
}
```
The user runs the following command to preview which data will be deleted.
```
aws ssm delete-inventory --type-name "Custom:RackSpace" --dry-run
```

The system returns information like the following.
```
{
  "DeletionId" : "1111-2222-333-444-66666",
  "DeletionSummary" : {
    "RemainingCount" : 3,
    "TotalCount" : 3,
    "SummaryItems" : [
      { "Count" : 2,
        "Version" : "1.0"
      },
      { "Count" : 1,
        "Version" : "2.0"
      }
    ]
  }
}
```

*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.*

Neither item was deleted.

The system found two items that use SchemaVersion 1.0.

This item was not deleted.

The system found one item that uses SchemaVersion 1.0.
The user runs the following command to delete the Custom:RackSpace inventory.

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

The system returns information like the following.

```
{
   "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"
}
```

**Viewing inventory delete actions in CloudWatch Events**

You can configure Amazon CloudWatch Events to create an event anytime a user deletes custom inventory. CloudWatch Events offers three types of events for custom inventory delete operations:

- **Delete action for an instance**: If the custom inventory for a specific managed instance was successfully deleted or not.
- **Delete action summary**: A summary of the delete action.
- **Warning for disabled custom inventory type**: A warning event if a user called the `PutInventory` API action for a custom inventory type version that was previously-disabled.

Here are examples of each event:

**Delete action for an instance**

```
{
   "version": "0",
   "id": "98c9cde-56c0-b38b-707f-0411b3ff9d11",
   "detail-type": "Inventory Resource State Change",
   "source": "aws.ssm",
```
Delete action summary

```json
{
  "version": "0",
  "id": "83898300-f576-5181-7a67-fb3e45e4fad4",
  "detail-type": "Inventory Resource State Change",
  "source": "aws.ssm",
  "account": "478678815555",
  "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":[]}",
    "type-name": "Custom:MyInfo"
  }
}
```

Warning for disabled custom inventory type

```json
{
  "version": "0",
  "id": "49c1855c-9c57-b5d7-8518-b64aeef5e4a",
  "detail-type": "Inventory Resource State Change",
  "source": "aws.ssm",
  "account": "478678815555",
  "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"
  }
}
```
Use the following procedure to create a CloudWatch Events rule for custom inventory delete operations. This procedure shows you how to create a rule that sends notifications for custom inventory delete operations to an Amazon SNS topic. Before you begin, verify that you have an Amazon SNS topic, or create a new one. For more information, see Getting Started in the Amazon Simple Notification Service Developer Guide.

To configure CloudWatch Events for delete inventory operations

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then choose Create rule.
3. Under Event Source, verify that Event Pattern is selected.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM).
5. In the Event Type field, choose Inventory.
6. Verify that Any detail type is selected, and then choose Add targets.
7. In the Select target type list, choose SNS topic, and then choose your topic from the list.
8. In the Configure input list, verify that Matched event is selected.
10. Specify a name and a description, and then choose Create rule.

Viewing inventory history and change tracking

You can view Systems Manager Inventory history and change tracking for all of your managed instances by using AWS Config. AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time. To view inventory history and change tracking, you must enable the following resources in AWS Config.

- SSM:ManagedInstanceInventory
- SSM:PatchCompliance
- SSM:AssociationCompliance

Note
By enabling SSM:PatchCompliance and SSM:AssociationCompliance, you can view Patch Manager patching and State Manager association compliance history and change tracking. For more information about compliance management for these resources, see Working with Configuration Compliance (p. 699).

The following procedure describes how to enable inventory history and change-track recording in AWS Config by using the AWS CLI. For more information about how to choose and configure these resources in AWS Config, see Selecting Which Resources AWS Config Records in the AWS Config Developer Guide. For information about AWS Config pricing, see Pricing.

Before You Begin

AWS Config requires AWS Identity and Access Management (IAM) permissions to get configuration details about Systems Manager resources. In the following procedure, you must specify an Amazon Resource Name (ARN) for an IAM role that gives AWS Config permission to Systems Manager resources. You can attach the AWSConfigRole managed policy to the IAM role that you assign to AWS Config. For information about how to create an IAM role and assign the AWSConfigRole managed policy to that role, see Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide.
To enable inventory history and change-track recording in AWS Config

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Copy and paste the following JSON sample into a simple text file and save it as recordingGroup.json.
   
   ```json
   {
     "allSupported":false,
     "includeGlobalResourceTypes":false,
     "resourceTypes": ["AWS::SSM::AssociationCompliance", "AWS::SSM::PatchCompliance", "AWS::SSM::ManagedInstanceInventory"]
   }
   ```
3. Run the following command to load the recordingGroup.json file into AWS Config.
   
   ```bash
   aws configservice put-configuration-recorder --configuration-recorder
   name=myRecorder,roleARN=arn:aws:iam::123456789012:role/myConfigRole --recording-group
   file://recordingGroup.json
   ```
4. Run the following command to start recording inventory history and change tracking.
   
   ```bash
   aws configservice start-configuration-recorder --configuration-recorder-name myRecorder
   ```

After you configure history and change tracking, you can drill down into the history for a specific managed instance by choosing the **AWS Config** button in the Systems Manager console.

You can access the **AWS Config** button from either the **Managed Instances** page or the **Inventory** page. Depending on your monitor size, you might need to scroll to the right side of the page to see the button.

## Systems Manager Inventory walkthroughs

Use the following walkthroughs to collect and manage inventory data. We recommend that you initially perform these walkthroughs with managed instances in a test environment.

### Before You Begin

Before you start these walkthroughs, complete the following tasks.

- Update SSM Agent on the instances you want to inventory. By running the latest version of SSM Agent, you ensure that you can collect metadata for all supported inventory types. For information about how to update SSM Agent by using State Manager, see Automatically update SSM Agent (CLI) (p. 908).
- Verify that your instances meet Systems Manager prerequisites. For more information, see Systems Manager prerequisites (p. 14).
- (Optional) Create a JSON file to collect custom inventory. For more information, see Working with custom inventory (p. 736).

### Contents

- Walkthrough: Assign custom inventory metadata to an instance (p. 748)
- Walkthrough: Configure your managed instances for Inventory by using the CLI (p. 749)
Walkthrough: Assign custom inventory metadata to an instance

The following procedure walks you through the process of using the PutInventory API action to assign custom inventory metadata to a managed instance. This example assigns rack location information to an instance. For more information about custom inventory, see Working with custom inventory (p. 736).

To assign custom inventory metadata to an instance

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to assign rack location information to an instance.

   ```sh
   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"}]'
   ```

3. Run the following command to view custom inventory entries for this instance.

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

   The system responds with information like the following.

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

4. Run the following command to view the custom inventory schema.

   ```sh
   aws ssm get-inventory-schema --type-name Custom:RackInfo
   ```

   The system responds with information like the following.

   ```json
   {
     "Schemas": [
       {
         "TypeName": "Custom:RackInfo",
         "Version": "1.0",
         "Attributes": [
           {
             "DataType": "STRING",
             "Name": "RackLocation"
           }
         ]
       }
     ]
   }
   ```
**Walkthrough: Configure your managed instances for Inventory by using the CLI**

The following procedures walk you through the process of configuring Systems Manager Inventory to collect metadata from your managed instances. When you configure inventory collection, you start by creating a Systems Manager State Manager association. Systems Manager collects the inventory data when the association is run. If you don't create the association first, and attempt to invoke the aws:softwareInventory plugin by using, for example, Run Command, the system returns the following error:

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

**Note**

An instance can have only one inventory association configured at a time. If you configure an instance with two or more inventory associations, the association doesn't run and no inventory data is collected.

**Quickly configure all of your managed instances for Inventory (CLI)**

You can quickly configure all managed instances in your AWS account and in the current Region to collect inventory data. This is called creating a global inventory association. To create a global inventory association by using the AWS CLI, use the wildcard option for the `instanceIds` value, as shown in the following procedure.

**To configure inventory for all managed instances in your AWS account and in the current Region (CLI)**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command.

   ```bash
   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,networkConfig=Enabled,services=Enabled,windowsRoles=Enabled,windowsUpdates=Enabled
   ```

   **Note**

   This command does not enable inventory of file or Windows Registry metadata. To inventory these datatypes, use the next procedure.

**Manually configuring Inventory on your managed instances (CLI)**

Use the following procedure to manually configure Systems Manager Inventory on your managed instances by using instance IDs or tags.

**To manually configure your managed instances for inventory (CLI)**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create a State Manager association that runs Systems Manager Inventory on the instance. This command configures the service to run every six hours and to collect network configuration, Windows Update, and application metadata from an instance.

   ```bash
   aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets
   "Key=instanceIds,Values=an instance ID" --schedule-expression "rate(240 minutes)"
   --output-location "{ "S3Location": { "OutputS3Region": "region-id" },\n   "OutputS3Bucket": "bucket-name" }" --parameters
   applications=Enabled,awsComponents=Enabled,customInventory=Enabled,instanceDetailedInformation=Enabled,networkConfig=Enabled,services=Enabled,windowsRoles=Enabled,windowsUpdates=Enabled
   ```
region-id represents the AWS Region where the instance is located, such as us-east-2 for the US East (Ohio) Region.

The system responds with information like the following.

```
{
  "AssociationDescription": {
    "ScheduleExpression": "rate(240 minutes)",
    "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"
      }
    ]
  }
}
```

You can target large groups of instances by using the Targets parameter with EC2 tags. For example:

```
aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets "Key=tag:Environment,Values=Production" --schedule-expression "rate(240 minutes)" --output-location "{ "S3Location": { "OutputS3Region": "us-east-2", "OutputS3BucketName": "Test bucket", "OutputS3KeyPrefix": "Test" } }" --parameters "networkConfig=Enabled, windowsUpdates=Enabled, applications=Enabled"
```

You can also inventory files and Windows Registry keys on a Windows Server instance by using the files and windowsRegistry inventory types with expressions. For more information about these inventory types, see Working with file and Windows registry inventory (p. 711).
aws ssm create-association --name "AWS-GatherSoftwareInventory" --targets "Key=instanceids,Values=i-0704358e3a3da9eb1" --schedule-expression "rate(240 minutes)" --parameters '{"files":[{"Path": "C:\Program Files", "Pattern": ["*.exe"], "Recursive": true}]]', "windowsRegistry": [{"Path": "HKEY_LOCAL_MACHINE\Software\Amazon", "Recursive": true}], "InstanceId": "i-0704358e3a3da9eb1"}' --profile dev-pdx

3. Run the following command to view the association status.

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

The system responds with information like the following.

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

Walkthrough: Use Resource Data Sync to aggregate inventory data

The following walkthrough describes how to create a resource data sync configuration by using the AWS CLI. A resource data sync automatically ports inventory data from all of your managed instances to a central S3 bucket. The sync automatically updates the data in the central S3 bucket whenever new inventory data is discovered. This walkthrough also describes how to use Amazon Athena and Amazon QuickSight to query and analyze the aggregated data. For information about creating a resource data sync by using the Amazon EC2 console, see Configuring Resource Data Sync for Inventory (p. 712).

**Note**
This walkthrough includes information about how to encrypt the sync by using AWS Key Management Service (AWS KMS). Inventory does not collect any user-specific, proprietary, or sensitive data so encryption is optional. For more information about AWS KMS, see AWS Key Management Service Developer Guide.

**Before You Begin**

Before you start this walkthrough, you must collect inventory metadata from your managed instances. For the purpose of the Amazon Athena and Amazon QuickSight sections in this walkthrough, we recommend that you collect Application metadata. For more information about how to collect inventory metadata, see Walkthrough: Configure your managed instances for Inventory by using the CLI (p. 749).

(Optional) If you want to encrypt the sync by using AWS KMS, then you must either create a new key that includes the following policy, or you must update an existing key and add this policy to it.

```
{
  "Version": "2012-10-17",
  "Id": "ssm-access-policy",
  "Statement": [
    ...
  ]
}
```
To create a Resource Data Sync for Inventory

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket to store your aggregated inventory data. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. Make a note of the bucket name and the AWS Region where you created it.
3. After you create the bucket, choose the Permissions tab, and then choose Bucket Policy.
4. Copy and paste the following bucket policy into the policy editor. Replace bucket-name and account-id with the name of the S3 bucket you created and a valid AWS account ID. Optionally, replace bucket-prefix with the name of an Amazon S3 prefix (subdirectory). If you did not create a prefix, remove bucket-prefix/ from the ARN in the policy.

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

5. (Optional) If you want to encrypt the sync, then you must add the following policy to the bucket. Repeat the previous step to add the following policy to the bucket.

   ```json
   { 
   "Version": "2012-10-17",
   "Statement": [
   
   ```
6. Install and configure the AWS CLI, if you have not already.

For information, see Install or upgrade AWS command line tools (p. 57).

7. (Optional) If you want to encrypt the sync, run the following command to verify that the bucket policy is enforcing the KMS key requirement.

```bash
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
```

8. Run the following command to create a resource data sync configuration with the S3 bucket you created at the start of this procedure. This command creates a sync from the AWS Region you are currently logged into.

   **Note**

   If the sync and the target S3 bucket are located in different regions, you may be subject to data transfer pricing. For more information, see Amazon S3 Pricing.

```bash
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"
```

You can use the `Region` parameter to specify where the sync configuration should be created. In the following example, inventory data from the us-west-1 Region, will be synchronized in the S3 bucket in the us-west-2 Region.

```bash
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
```

(Optional) If you want to encrypt the sync by using AWS KMS, run the following command to create the sync. If you encrypt the sync, then the AWS KMS key and the S3 bucket must be in the same Region.

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

9. Run the following command to view the status of sync configuration.

```bash
aws ssm list-resource-data-sync
```
If you created the sync configuration in a different Region, then you must specify the `region` parameter, as shown in the following example.

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

10. After the sync configuration is created successfully, browse the target bucket in Amazon S3. Inventory data should appear within a few minutes.

**Working with the Data in Amazon Athena**

The following section describes how to view and query the data in Amazon Athena. Before you begin, we recommend that you learn about Athena. For more information, see What is Amazon Athena? and Working with Data in the Amazon Athena User Guide.

**To view and query the data in Amazon Athena**

2. Copy and paste the following statement into the query editor and then choose **Run Query**.

```
CREATE DATABASE ssminventory
```

The system creates a database called ssminventory.

3. Copy and paste the following statement into the query editor and then choose **Run Query**. Replace `bucket-name` and `bucket-prefix` with the name and prefix of the Amazon S3 target.

```
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. Copy and paste the following statement into the query editor and then choose **Run Query**.

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

The system partitions the table.

**Note**

If you create resource data syncs from additional AWS Regions or accounts, then you must run this command again to update the partitions. You may also need to update your S3 bucket policy.

5. To preview your data, choose the view icon next to the AWS_Application table.
6. Copy and paste the following statement into the query editor and then choose **Run Query**.

```sql
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
```

The query returns a count of different versions of `aws-cfn-bootstrap`, which is an AWS application present on EC2 instances for both Linux and Windows Server.

7. Individually copy and paste the following statements into the query editor, replace `bucket-name` and `bucket-prefix` with information for Amazon S3, and then choose **Run Query**. These statements set up additional inventory tables in Athena.

```sql
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,
    `755`
`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,
`OperationStart` string,
`OperationEnd` 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

Working with the Data in Amazon QuickSight

The following section provides an overview with links for building a visualization in Amazon QuickSight.
To build a visualization in Amazon QuickSight

1. Sign up for Amazon QuickSight and then log in to the QuickSight console.
2. Create a data set from the AWS_Application table and any other tables you created. For more information, see Creating a Data Set Using Amazon Athena Data.
3. Join tables. For example, you could join the instanceid column from AWS_InstanceInformation because it matches the resourceid column in other inventory tables. For more information about joining tables, see Joining Tables.
4. Build a visualization. For more information, see Working with Amazon QuickSight Visuals.

Troubleshooting problems with Systems Manager Inventory

This topic includes information about how to troubleshoot common errors or problems with Systems Manager Inventory.

Topics
- The AWS-ListWindowsInventory document fails to run (p. 757)
- Console doesn't display Inventory Dashboard | Detailed View | Settings tabs (p. 757)
- UnsupportedAgent (p. 758)
- Skipped (p. 758)
- Failed (p. 758)

The AWS-ListWindowsInventory document fails to run

The AWS-ListWindowsInventory document is deprecated. Don't use this document to collect inventory. Instead, use one of the processes described in Configuring inventory collection (p. 718).

Console doesn't display Inventory Dashboard | Detailed View | Settings tabs

The Inventory Detailed View page is only available in AWS Regions that offer Amazon Athena. If the following tabs are not displayed on the Inventory page, it means Athena is not available in the Region and you can't use the Detailed View to query data.
UnsupportedAgent

If the detailed status of an inventory association shows **UnsupportedAgent**, and the **Association status** shows **Failed**, then the version of SSM Agent on the instance is not correct. To create a global inventory association (to inventory all instances in your AWS account) for example, you must use SSM Agent version 2.0.790.0 or later. You can view the agent version running on each of your instances on the **Managed Instances** page in the **Agent version** column. For information about how to update SSM Agent on your instances, see Update SSM Agent by using Run Command (p. 835).

Skipped

If the status of the inventory association for an instance shows **Skipped**, this means that you created a global inventory association, but the skipped instance already had an inventory association assigned to it. The global inventory association was not assigned to this instance, and no inventory was collected by the global inventory association. However, the instance will still report inventory data when the specific inventory association runs.

Failed

If the status of the inventory association for an instance shows **Failed**, this could mean that the instance has multiple inventory associations assigned to it. An instance can only have one inventory association assigned at a time. An inventory association uses the AWS-GatherSoftwareInventory SSM document. You can run the following command by using the AWS CLI to view a list of associations for an instance.

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

AWS Systems Manager Managed Instances

A managed instance is any machine configured for AWS Systems Manager. You can configure EC2 instances or on-premises machines in a hybrid environment as managed instances. Systems Manager supports various distributions of Linux, including Raspberry Pi devices, and Microsoft Windows Server.

**Note**

In the AWS Management Console, any machine prefixed with "mi-" is an on-premises server or virtual machine (VM) managed instance.

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers and VMs in your hybrid environment. The standard-instances tier enables you to register a maximum of 1,000 servers or VMs per AWS account per AWS Region. If you need to register more than 1,000 servers or VMs in a single account and Region, then use the advanced-instances tier. You can create as many instances as you like in the advanced-instances tier, but all instances configured for Systems Manager are priced on a pay-per-use basis. For more information about enabling advanced instances, see Enabling the advanced-instances tier (p. 760). For more information about pricing, see AWS Systems Manager Pricing.

**Note**

- Advanced instances also enable you to connect to your hybrid machines by using AWS Systems Manager Session Manager. Session Manager provides interactive shell access to your instances. For more information, see AWS Systems Manager Session Manager (p. 770).
- The standard-instances quota also applies to EC2 instances that use a Systems Manager on-premises activation (which is not a common scenario).
- Microsoft application patching is only available on EC2 instances and in the advanced-instances tier. To patch Microsoft applications on on-premises servers and VMs, you must
enable the advanced-instances tier. For more information, see About patching applications on Windows Server (p. 957).

If you don’t see your managed instances listed in the console, then do the following:

1. Verify that the console is open in the AWS Region where you created your managed instances. You can switch Regions by using the list in the top, right corner of the console.
2. Verify that your instances meet Systems Manager requirements. For information, see Systems Manager prerequisites (p. 14).
3. For servers and VMs in a hybrid environment, verify that you completed the activation process. For more information, see Setting up AWS Systems Manager for hybrid environments (p. 43).

**Note**

Systems Manager requires accurate time references in order to perform its operations. If your instance’s date and time are not set correctly, they may not match the signature date of your API requests. For more information, see Use cases and best practices (p. 1199).

**Verify Systems Manager support on an instance**

AWS Config provides AWS Managed Rules, which are predefined, customizable rules that AWS Config uses to evaluate whether your AWS resource configurations comply with common best practices. AWS Config Managed Rules include the ec2-instance-managed-by-systems-manager rule. This rule checks whether the EC2 instances in your account are managed by Systems Manager. For more information, see AWS Config Managed Rules.

**Verify Systems Manager Prerequisites**

For information about Systems Manager prerequisites, see Systems Manager prerequisites (p. 14). For information about configuring on-premises servers and VMs as managed instances, see Setting up AWS Systems Manager for hybrid environments (p. 43).

**Increase security posture on managed instances**

For more information increasing your security posture against unauthorized root-level commands on your instances, see Restrict access to root-level commands through SSM Agent (p. 92)

**Topics**

- Configuring instance tiers (p. 759)
- Resetting passwords on managed instances (p. 766)
- Deregistering managed instances in a hybrid environment (p. 769)

**Configuring instance tiers**

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers and VMs in your hybrid environment. The standard-instances tier enables you to register a maximum of 1,000 on-premises servers or VMs per AWS account per AWS Region. If you need to register more than 1,000 on-premises servers or VMs in a single account and Region, then use the advanced-instances tier. You can activate as many managed instances in a hybrid environment as you like in the advanced-instances tier. However, all instances configured for Systems Manager using the managed-instance activation process described earlier in Step 4: Create a managed-instance activation for a hybrid environment (p. 48) are made available on a pay-per-use basis. This also applies to EC2 instances that use a Systems Manager on-premises activation (which is not a common scenario).

As long as your AWS account and Region has fewer than 1,000 on-premises instances in your hybrid environment, you can revert back to the standard-instances tier at any time.
Enabling the advanced-instances tier

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers and VMs in your hybrid environment. The standard-instances tier enables you to register a maximum of 1,000 on-premises servers or VMs per AWS account per AWS Region. If you need to register more than 1,000 on-premises servers or VMs in a single account and Region, then use the advanced-instances tier. You can activate as many managed instances in a hybrid environment as you like in the advanced-instances tier. However, all instances configured for Systems Manager using the managed-instance activation process described earlier in Create a managed-instance activation for a hybrid environment (p. 48) are made available on a pay-per-use basis. This also applies to EC2 instances that use a Systems Manager on-premises activation (which is not a common scenario).

Note

- Advanced instances also enable you to connect to your hybrid machines by using AWS Systems Manager Session Manager. Session Manager provides interactive shell access to your instances. For more information, see AWS Systems Manager Session Manager (p. 770).
- The standard-instances limit also applies to EC2 instances that use a Systems Manager on-premises activation (which is not a common scenario).
- Microsoft application patching is only available on EC2 instances and in the advanced-instances tier. To patch Microsoft applications on on-premises servers and VMs, you must enable the advanced-instances tier. For more information, see About patching applications on Windows Server (p. 957).

This section describes how to configure your hybrid environment to use the advanced-instances tier.

Before You Begin

Review pricing details for advanced instances. Advanced instances are an account-level feature. Advanced instances are available on a per-use-basis. For more information see, AWS Systems Manager Pricing.

Configuring permissions to enable the advanced-instances tier

Verify that you have permission in AWS Identity and Access Management (IAM) to change your environment from the standard-instances tier to the advanced-instances tier. You must either have the AdministratorAccess policy attached to your IAM user, group, or role. Or, you must have permission to change the Systems Manager activation-tier service setting. The activation-tier setting uses the following API actions:

- GetServiceSetting
- UpdateServiceSetting
- ResetServiceSetting

Use the following procedure to add an inline IAM policy to a user account. This policy enables a user to view the current managed-instance tier setting. This policy also enables the user to change or reset the current setting in the specified AWS account and Region.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Users.
3. In the list, choose the name of the user to embed a policy in.
4. Choose the Permissions tab.
5. On the right side of the page, under Permission policies, choose Add inline policy.
6. Choose the JSON tab.
7. Replace the default content with the following:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ssm:GetServiceSetting"
         ],
         "Resource": "*",
      },
      {
         "Effect": "Allow",
         "Action": [
            "ssm:ResetServiceSetting",
            "ssm:UpdateServiceSetting"
         ],
      }
   ]
}
```

9. On the Review policy page, for Name, enter a name for the inline policy. For example: Managed-Instances-Tier.
10. Choose Create policy.

Administrators can specify read-only permission by assigning the following inline policy to the user's account.

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ssm:GetServiceSetting"
         ],
         "Resource": "*",
      },
      {
         "Effect": "Deny",
         "Action": [
            "ssm:ResetServiceSetting",
            "ssm:UpdateServiceSetting"
         ],
         "Resource": "*",
      }
   ]
}
```
For more information about creating and editing IAM policies, see Creating IAM Policies in the IAM User Guide.

Enabling the advanced-instances tier (console)

The following procedure shows you how to use the Systems Manager console to change all on-premises servers and VMs that were added using managed-instance activation, in the specified AWS account and Region, to use the advanced-instances tier.

Important
The following procedure describes how to change an account-level setting. This change results in charges being billed to your account.

To enable the advanced-instances tier (console)

2. In the navigation pane, choose Managed instances.
3. Choose the Settings tab.

   If you don’t see the Settings tab, then do the following:
   1. Verify that the console is open in the AWS Region where you created your managed instances. You can switch Regions by using the list in the top, right corner of the console.
   2. Verify that your instances meet Systems Manager requirements. For information, see Systems Manager prerequisites (p. 14).
   3. For servers and VMs in a hybrid environment, verify that you completed the activation process. For more information, see Setting up AWS Systems Manager for hybrid environments (p. 43).
4. Choose Change account settings.
5. Review the information in the pop-up about changing account settings, and then, if you approve, choose the option to accept and continue.

The system can take several minutes to complete the process of moving all instances from the standard-instances tier to the advanced-instances tier.

Note
For information about changing back to the standard-instances tier, see Reverting from the advanced-instances tier to the standard-instances tier (p. 764).

Enabling the advanced-instances tier (AWS CLI)

The following procedure shows you how to use the AWS CLI to change all on-premises servers and VMs that were added using managed-instance activation, in the specified AWS account and Region, to use the advanced-instances tier.

Important
The following procedure describes how to change an account-level setting. This change results in charges being billed to your account.

To enable the advanced-instances tier using the AWS CLI

1. Open the AWS CLI and run the following command.

   ```bash
   ```

   There is no output if the command succeeds.
2. Run the following command to view the current service settings for managed instances in the current AWS account and Region.

```bash
```

```
{
  "ServiceSetting": {
    "SettingId": "/ssm/managed-instance/activation-tier",
    "SettingValue": "advanced",
    "LastModifiedDate": 1555603376.138,
    "LastModifiedUser": "arn:aws:sts::123456789012:assumed-role/Administrator/User_1",
    "Status": "PendingUpdate"
  }
}
```

### Enabling the advanced-instances tier (PowerShell)

The following procedure shows you how to use the AWS Tools for Windows PowerShell to change all on-premises servers and VMs that were added using managed-instance activation, in the specified AWS account and Region, to use the advanced-instances tier.

**Important**

The following procedure describes how to change an account-level setting. This change results in charges being billed to your account.

**To enable the advanced-instances tier using PowerShell**

1. Open AWS Tools for Windows PowerShell and run the following command.

```powershell
```

There is no output if the command succeeds.

2. Run the following command to view the current service settings for managed instances in the current AWS account and Region.

```powershell
```

```
LastModifiedDate : 4/18/2019 4:02:56 PM
LastModifiedUser : arn:aws:sts::123456789012:assumed-role/Administrator/User_1
SettingId : /ssm/managed-instance/activation-tier
SettingValue : advanced
Status : PendingUpdate
```

The system can take several minutes to complete the process of moving all instances from the standard-instances tier to the advanced-instances tier.
Configuring instance tiers

Reverting from the advanced-instances tier to the standard-instances tier

This section describes how to change hybrid instances currently running in the advanced-instances tier back to the standard-instances tier. This configuration applies to all hybrid instances in an AWS account and a single AWS Region.

Before You Begin

Review the following important details.

Note

- You can't revert back to the standard-instance tier if you are running more than 1,000 hybrid instances in the account and Region. You must first deregister hybrid instances until you have 1,000 or fewer. This also applies to EC2 instances that use a Systems Manager on-premises activation (which is not a common scenario). For more information, see Deregistering managed instances in a hybrid environment (p. 769).
- After you revert, you won't be able to use Session Manager to interactively access your hybrid instances.
- After you revert, you won't be able to use Patch Manager to patch Microsoft applications on hybrid servers and virtual machines (VMs).
- The process of reverting all hybrid instances back to the standard-instance tier can take 30 minutes or more to complete.

This section describes how to revert all hybrid instances in an AWS account and Region from the advanced-instances tier to the standard-instances tier.

Reverting to the standard-instances tier (console)

The following procedure shows you how to use the Systems Manager console to change all on-premises servers and VMs in your hybrid environment to use the standard-instances tier in the specified AWS account and Region.

To revert to the standard-instances tier (console)

2. In the navigation pane, choose Managed instances.
3. Choose the Settings tab.
4. Choose Change account setting.
5. Review the information in the pop-up about changing account settings, and then if you approve, choose the option to accept and continue.

Reverting to the standard-instances tier (AWS CLI)

The following procedure shows you how to use the AWS CLI to change all on-premises servers and VMs in your hybrid environment to use the standard-instances tier in the specified AWS account and Region.

To revert to the standard-instances tier using the AWS CLI

1. Open the AWS CLI and run the following command.

There is no output if the command succeeds.

2. Run the following command 30 minutes later to view the settings for managed instances in the current AWS account and Region.


The command returns information like the following.

```json
{
   "ServiceSetting": {
      "SettingId": "/ssm/managed-instance/activation-tier",
      "SettingValue": "standard",
      "LastModifiedDate": 1555603376.138,
      "LastModifiedUser": "System",
      "Status": "Default"
   }
}
```

The status changes to Default after the request has been approved.

**Reverting to the standard-instances tier (PowerShell)**

The following procedure shows you how to use AWS Tools for Windows PowerShell to change all on-premises servers and VMs in your hybrid environment to use the standard-instances tier in the specified AWS account and Region.

**To revert to the standard-instances tier using PowerShell**

1. Open AWS Tools for Windows PowerShell and run the following command.

```
```

There is no output if the command succeeds.

2. Run the following command 30 minutes later to view the settings for managed instances in the current AWS account and Region.

```
```

The command returns information like the following.

```
LastModifiedDate : 4/18/2019 4:02:56 PM
```
The status changes to Default after the request has been approved.

## Resetting passwords on managed instances

You can reset the password for any user on a managed instance. This includes Amazon EC2 instances, on-premises servers, and virtual machines (VMs) that are managed by AWS Systems Manager. The password reset functionality is built on the AWS Systems Manager Session Manager capability. You can use this functionality to connect to instances without opening inbound ports, maintaining bastion hosts, or managing SSH keys.

This makes the password reset option useful when a user has forgotten a password, or when you want to quickly update a password without making an RDP or SSH connection to the instance.

### Prerequisites

Before you can reset the password on an instance, the following requirements must be met:

- The instance you want to change a password on must be a Systems Manager managed instance. This means that SSM Agent is installed on the instance. (SSM Agent Version 2.3.668.0 or later is required for changing passwords.) For information about installing or updating SSM Agent, see Working with SSM Agent (p. 63).
- The password reset functionality uses the AWS Session Manager configuration that is set up for your account to connect to the instance. Therefore, the prerequisites for using Session Manager must have been completed for your account in the current Region. For more information, see Getting started with Session Manager (p. 773).
  
  **Note**  
  Session Manager support for on-premises servers is provided for the advanced-instances tier only. For information, see Enabling the advanced-instances tier (p. 760).
- The AWS user who is changing the password must have the `ssm:SendCommand` permission for the instance. For information, see Restricting Run Command access based on instance tags (p. 831).

### Restricting Access

You can limit a user's ability to reset passwords to specific instances. This is done by using identity-based policies for the Session Manager `ssm:StartSession` action with the `AWS-PasswordReset SSM` document. For more information, see Control user session access to instances (p. 782).

### Encrypting Data

You must enable AWS Key Management Service (AWS KMS) end-to-end encryption for Session Manager data to use the password reset option for managed instances. For more information, see Enable AWS KMS key encryption of session data (console) (p. 799).

### Reset a password on a managed instance

You can reset a password on a Systems Manager managed instance using the AWS Systems Manager Managed Instances console or the AWS CLI.

**To change the password on a managed instance (console)**

2. In the navigation pane, choose **Managed Instances**.

-or-

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Managed Instances**.

3. Choose the button next to the instance that needs a new password.

4. In the **Actions** menu, choose **Reset password**.

5. For **User name**, type the name of the user for which you are changing the password. This can be any user name that has an account on the instance.

6. Choose **Submit**.

7. Follow the prompts in the **Enter new password** command window to specify the new password.

   **Note**
   If the version of SSM Agent on the instance doesn't support password resets, you are prompted to install a supported version using Run Command.

**To reset the password on a managed instance (CLI)**

1. To reset the password for a user on a managed instance, run the following command.

   **Note**
   To use the AWS CLI to reset a password, the Session Manager plugin must be installed on your local machine. For information, see *(Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).*

   ```
   aws ssm start-session --target instance-id --document-name "AWS-PasswordReset" --
   parameters "{"username": "user-name"}"
   ```

   **instance-id** represents the ID of an instance configured for use with Systems Manager and its Session Manager capability.

   **user-name** represents the name of the user you want to reset password for on the instance.

2. Follow the prompts in the **Enter new password** command window to specify the new password.

**Troubleshoot password resets on managed instances**

Many password reset issues can be resolved by ensuring that you have completed the password reset prerequisites (p. 766). For other problems, use the following information to help you troubleshoot password reset issues.

**Topics**

- **Instance not available** (p. 767)
- **SSM Agent not up-to-date (console)** (p. 768)
- **Password reset options do not appear (CLI)** (p. 768)
- **No authorization to run ssm:SendCommand** (p. 768)
- **Session Manager error message** (p. 768)

**Instance not available**

**Problem**: You want to reset the password for an EC2 instance on the **Managed instances** console page, but the instance is not in the list.
• **Solution**: The instance you want to connect to might not be configured to use with the AWS Systems Manager service. To use an EC2 instance with Systems Manager, an IAM instance profile that gives Systems Manager permission to perform actions on your instances must be attached to the instance. For information, see Create an IAM instance profile for Systems Manager (p. 50). To use an on-premises server or virtual machine (VM) that you have activated for use with Systems Manager, you must create an IAM service role that gives Systems Manager permission to perform actions on your machines. For information, see Create an IAM service role for a hybrid environment (p. 44). (Session Manager support for on-premises servers and VMs is provided for the advanced-instances tier only. For information, see Enabling the advanced-instances tier (p. 760).)

**SSM Agent not up-to-date (console)**

**Problem**: A message reports that the version of SSM Agent doesn't support password reset functionality.

• **Solution**: Version 2.3.668.0 or later of SSM Agent is required to perform password resets. In the console, you can begin the process of updating the agent on the instance by choosing **Update SSM Agent**.

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

**Password reset options do not appear (CLI)**

**Problem**: You connect successfully to an instance using the AWS CLI `start-session` command. You specified the SSM Document `AWS-PasswordReset` and provided a valid user name, but prompts to change the password do not appear.

• **Solution**: The version of SSM Agent on the instance is not up-to-date. Version 2.3.668.0 or later is required to perform password resets.

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

**No authorization to run `ssm:SendCommand`**

**Problem**: You attempt to connect to an instance to change its password but receive an error message saying that you aren’t authorized to run `ssm:SendCommand` on the instance.

• **Solution**: Your IAM user policy must include permission to run the `ssm:SendCommand` command. For information, see Restricting Run Command access based on instance tags (p. 831).

**Session Manager error message**

**Problem**: You receive an error message related to Session Manager.
Deregistering managed instances in a hybrid environment

If you no longer want to manage an on-premises server or virtual machine (VM) by using AWS Systems Manager, then you can deregister it. Deregistering a hybrid machine removes it from the list of managed instances in Systems Manager. SSM Agent running on the hybrid machine won't be able to refresh its authorization token because it's no longer registered. SSM Agent will hibernate and reduce its ping frequency to Systems Manager in the cloud to once per hour.

You can reregister an on-premises server or VM again at any time. Systems Manager stores the command history for a deregistered managed instance for 30 days.

The following procedure describes how to deregister a hybrid machine by using the AWS Systems Manager console. For information about how to do this by using the AWS CLI, see deregister-managed-instance.

To deregister a hybrid machine (console)

2. In the navigation pane, choose Managed instances.
3. In the Managed instances list, choose the instance you want to deregister and then choose Actions, Deregister this managed instance.
4. Review the information in the Deregister this managed instance pop-up, and then if you approve, choose Deregister.

AWS Systems Manager hybrid activations

To set up servers and virtual machines (VMs) in your hybrid environment as managed instances, you create a managed-instance hybrid activation. After you complete the activation, you receive an activation code and ID. This code/ID combination functions like an Amazon EC2 access ID and secret key to provide secure access to the Systems Manager service from your managed instances.

For information about configuring on-premises servers and VMs as managed instances, see Setting up AWS Systems Manager for hybrid environments (p. 43).

About AWS Systems Manager Instances Tiers

AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers and VMs in your hybrid environment. The standard-instances tier enables you to register a maximum of 1,000 servers or VMs per AWS account per AWS Region. If you need to register more than 1,000 servers or VMs in a single account and Region, then use the advanced-instances tier. You can create as many instances as you like in the advanced-instances tier, but all instances configured for Systems Manager are priced on a pay-per-use basis. For more information about enabling advanced instances, see Enabling the advanced-instances tier (p. 760). For more information about pricing, see AWS Systems Manager Pricing.

Note

- Advanced instances also enable you to connect to your hybrid machines by using AWS Systems Manager Session Manager. Session Manager provides interactive shell access to your instances. For more information, see AWS Systems Manager Session Manager (p. 770).
AWS Systems Manager Session Manager

Session Manager is a fully managed AWS Systems Manager capability that lets you manage your EC2 instances, on-premises instances, and virtual machines (VMs) through an interactive one-click browser-based shell or through the AWS CLI. Session Manager provides secure and auditable instance management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your managed instances.

How can Session Manager benefit my organization?

Session Manager offers these benefits:

- **Centralized access control to instances using IAM policies**
  Administrators have a single place to grant and revoke access to instances. Using only AWS Identity and Access Management (IAM) policies, you can control which individual users or groups in your organization can use Session Manager and which instances they can access.

- **No open inbound ports and no need to manage bastion hosts or SSH keys**
  Leaving inbound SSH ports and remote PowerShell ports open on your instances greatly increases the risk of entities running unauthorized or malicious commands on the instances. Session Manager helps you improve your security posture by letting you close these inbound ports, freeing you from managing SSH keys and certificates, bastion hosts, and jump boxes.

- **One-click access to instances from the console and CLI**
  Using the AWS Systems Manager console or Amazon EC2 console, you can start a session with a single click. Using the AWS CLI, you can also start a session that runs a single command or a sequence of commands. Because permissions to instances are provided through IAM policies instead of SSH keys or other mechanisms, the connection time is greatly reduced.

- **Port forwarding**
  Redirect any port inside your remote instance to a local port on a client. After that, connect to the local port and access the server application that is running inside the instance.

- **Cross-platform support for both Windows and Linux**
  Session Manager provides both Windows and Linux support from a single tool. For example, you don't need to use an SSH client for Linux instances and an RDP connection for Windows Server instances.

- **Logging and auditing session activity**
  To meet operational or security requirements in your organization, you might need to provide a record of the connections made to your instances and the commands that were run on them. You can also receive notifications when a user in your organization starts or ends session activity.

Logging and auditing capabilities are provided through integration with the following AWS services:

- **AWS CloudTrail** – AWS CloudTrail captures information about Session Manager API calls made in your AWS account and writes it to log files that are stored in an S3 bucket you specify. One bucket
Who should use Session Manager?

- Any AWS customer who wants to improve their security and audit posture, reduce operational overhead by centralizing access control on instances, and reduce inbound instance access.
- Information Security experts who want to monitor and track instance access and activity, close down inbound ports on instances, or enable connections to instances that do not have a public IP address.
- Administrators who want to grant and revoke access from a single location, and who want to provide one solution to users for both Windows and Linux instances.
- End users who want to connect to an instance with just one click from the browser or CLI without having to provide SSH keys.

What are the main features of Session Manager?

- **Support for both Windows Server and Linux instances**

  Session Manager lets you establish secure connections to your Amazon Elastic Compute Cloud (EC2) instances, on-premises instances, and virtual machines (VMs). For a list of supported Windows and Linux operating system types, see Getting started with Session Manager (p. 773).

  **Note**

  Session Manager support for on-premises servers is provided for the advanced-instances tier only. For information, see Enabling the advanced-instances tier (p. 760).

- **Console, CLI, and SDK access to Session Manager capabilities**

  You can work with Session Manager in the following ways:

  The **AWS Systems Manager console** includes access to all the Session Manager capabilities for both administrators and end-users. You can perform any task that’s related to your sessions by using the Systems Manager console.
The Amazon EC2 console provides the ability for end-users to connect to the EC2 instances for which they have been granted session permissions.

The AWS CLI includes access to Session Manager capabilities for end users. You can start a session, view a list of sessions, and permanently end a session by using the AWS CLI.

Note
To use the AWS CLI to run session commands, you must be using version 1.16.12 of the CLI (or later), and you must have installed the Session Manager plugin on your local machine. For information, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

The Session Manager SDK consists of libraries and sample code that enables application developers to build frontend applications, such as custom shells or self-service portals for internal users that natively use Session Manager to connect to instances. Developers and partners can integrate Session Manager into their client-side tooling or Automation workflows using the Session Manager APIs. You can even build custom solutions.

- **IAM access control**

  Through the use of IAM policies, you can control which members of your organization can initiate sessions to instances and which instances they can access. You can also provide temporary access to your instances. For example, you might want to give an on-call engineer (or a group of on-call engineers) access to production servers only for the duration of their rotation.

- **Logging and auditing capability support**

  Session Manager provide you with options for auditing and logging session histories in your AWS account through integration with a number of other AWS services. For more information, see Auditing and logging session activity (p. 823).

- **Customer key data encryption support**

  You can configure Session Manager to encrypt the session data logs that you send to an S3 bucket or stream to a CloudWatch Logs log group. You can also configure Session Manager to further encrypt the data transmitted between client machines and your instances during your sessions. For information, see Auditing and logging session activity (p. 823) and Configure session preferences (p. 796).

- **AWS PrivateLink support for instances without public IP addresses**

  You can also set up VPC Endpoints for Systems Manager using AWS PrivateLink to further secure your sessions. PrivateLink limits all network traffic between your managed instances, Systems Manager, and Amazon EC2 to the Amazon network. For more information, see (Optional) Create a Virtual Private Cloud endpoint (p. 37).

- **Tunneling**

  In a session, use a Session-type SSM document to tunnel traffic, such as http or a custom protocol, between a local port on a client machine and a remote port on an instance.

- **Interactive Commands**

  Create a Session-type SSM document that uses a session to interactively run a single command, giving you a way to manage what users can do on an instance.

What is a session?

A session is a connection made to an instance using Session Manager. Sessions are based on a secure bi-directional communication channel between the client (you) and the remote managed instance that streams inputs and outputs for commands. Traffic between a client and a managed instance is encrypted using TLS 1.2, and requests to create the connection are signed using Sigv4. This two-way...
communication enables interactive bash and PowerShell access to instances. You can also use an AWS Key Management Service (AWS KMS) key to further encrypt data beyond the default TLS encryption.

For example, say that John is an on-call engineer in your IT department. He receives notification of an issue that requires him to remotely connect to an instance, such as a failure that requires troubleshooting or a directive to change a simple configuration option on an instance. Using the AWS Systems Manager console, the Amazon EC2 console, or the AWS CLI, John starts a session connecting him to the instance, runs commands on the instance needed to complete the task, and then ends the session.

When John sends that first command to start the session, the Session Manager service authenticates his ID, verifies the permissions granted to him by an IAM policy, checks configuration settings (such as verifying allowed limits for the sessions), and sends a message to SSM Agent to open the two-way connection. After the connection is established and John types the next command, the command output from SSM Agent is uploaded to this communication channel and sent back to his local machine.

Topics
- Getting started with Session Manager (p. 773)
- Working with Session Manager (p. 812)
- Auditing and logging session activity (p. 823)
- Troubleshooting Session Manager (p. 827)

Getting started with Session Manager

Before you use Session Manager to connect to the EC2 instances in your account, complete the steps in the following topics.

Topics
- Step 1: Complete Session Manager prerequisites (p. 773)
- Step 2: Verify or create an IAM instance profile with Session Manager permissions (p. 776)
- Step 3: Control user session access to instances (p. 782)
- Step 4: Configure session preferences (p. 796)
- Step 5: (Optional) Restrict access to commands in a session (p. 804)
- Step 6: (Optional) use PrivateLink to set up a VPC endpoint for Session Manager (p. 808)
- Step 7: (Optional) disable or enable ssm-user account administrative permissions (p. 808)
- Step 8: (Optional) Enable SSH connections through Session Manager (p. 810)

Step 1: Complete Session Manager prerequisites

Before using Session Manager, make sure your environment meets the following requirements.

Session Manager prerequisites

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Operating Systems</td>
<td>AWS Session Manager supports the following operating system versions:</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td>Session Manager supports EC2 instances, as well as servers or virtual machines (VMs) in your hybrid environment that use the <em>advanced-instances</em> tier. For more</td>
</tr>
<tr>
<td>Requirement</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>information about advanced instances, see [Enabling the advanced-instances tier](p. 760).</td>
</tr>
</tbody>
</table>

**Linux**

Session Manager supports all the versions of Linux that are supported for AWS Systems Manager as a whole. For information, see [Systems Manager prerequisites](p. 14).

**Windows**


**Note**

Microsoft Windows Server 2016 Nano is not supported.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM Agent</td>
<td>SSM Agent version 2.3.68.0 or later must be installed on the instances you want to connect to through sessions. To use the option to encrypt session data using a customer master key (CMK) created in AWS Key Management Service (AWS KMS), version 2.3.539.0 or later of SSM Agent must be installed.</td>
</tr>
<tr>
<td></td>
<td>To install or update SSM Agent, see Working with SSM Agent (p. 63).</td>
</tr>
</tbody>
</table>

**About the ssm-user account**

Starting with version 2.3.50.0 of SSM Agent, the agent creates a user account on the instance, with root or administrator privileges, called `ssm-user`. (On versions before 2.3.612.0, the account is created when SSM Agent starts or restarts. On version 2.3.612.0 and later, `ssm-user` is created the first time a session starts on the instance.) Sessions are launched using the administrative credentials of this user account. For information about restricting administrative control for this account, see Step 7: (Optional) disable or enable ssm-user account administrative permissions (p. 808).

**Ssm-user on Windows Server domain controllers**

Beginning with SSM Agent version 2.3.612.0, the `ssm-user` account is not created automatically on managed instances that are used as Windows Server domain controllers. To use Session Manager on a Windows Server machine being used as a domain controller, you must create the `ssm-user` account manually if it isn't already present. On Windows Server, SSM Agent sets a new password for the `ssm-user` account each time a session starts, so you do not need to specify a password when you create the account.
Getting started with Session Manager

### Requirement Description

**AWS CLI**

(Optional) If you use the AWS CLI to start your sessions (instead of using the AWS Systems Manager console or Amazon EC2 console), version 1.16.12 or later of the CLI must be installed on your local machine.

You can call `aws --version` to check the version.

If you need to install or upgrade the CLI, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide.

**Important**

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

In addition, to use the CLI to manage your instances with Session Manager, you must first install the Session Manager plugin on your local machine. For information, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

---

**Step 2: Verify or create an IAM instance profile with Session Manager permissions**

By default, AWS Systems Manager doesn’t have permission to perform actions on your instances. You must grant access by using an IAM instance profile. An instance profile is a container that passes IAM role information to an EC2 instance at launch. This requirement applies to permissions for all AWS Systems Manager capabilities, not only those specific to Session Manager.

If you already use other Systems Manager capabilities, such as Run Command or Parameter Store, an instance profile with the required basic permissions for Session Manager might already be attached to your instances. If an instance profile that contains the AWS managed policy `AmazonSSMManagedInstanceCore` is already attached to your instances, the required permissions for Session Manager are already provided.

However, in some cases, you might need to modify the permissions attached to your instance profile. For example, you want to provide a narrower set of instance permissions, you have created a custom policy for your instance profile, or you want to use Amazon S3 encryption or AWS KMS encryption options for securing session data. For these cases, do one of the following to allow Session Manager actions to be performed on your instances:
• Embed permissions for Session Manager actions in a custom instance profile

To add permissions for Session Manager actions to an existing IAM instance profile that does not rely on the AWS-provided default policy `AmazonSSMManagedInstanceCore`, follow the steps in Adding Session Manager permissions to an existing instance profile (p. 777).

• Create a custom IAM instance profile with Session Manager permissions only

To create an IAM instance profile that contains permissions only for Session Manager actions, follow the steps in Create a custom IAM instance profile for Session Manager (p. 778).

• Create and use a new instance profile with permissions for all Systems Manager actions

To create an IAM instance profile for Systems Manager managed instances that uses a default policy supplied by AWS to grant all Systems Manager permissions, follow the steps in Create an IAM instance profile for Systems Manager (p. 30).

Note
You can attach an IAM instance profile to an EC2 instance as you launch it or to a previously launched instance. For more information, see Instance Profiles.

Topics
• Adding Session Manager permissions to an existing instance profile (p. 777)
• Create a custom IAM instance profile for Session Manager (p. 778)

Adding Session Manager permissions to an existing instance profile

Follow these steps to embed Session Manager permissions in an existing IAM instance profile that does not rely on the AWS-provided default policy `AmazonSSMManagedInstanceCore` for instance permissions. Note that this procedure assumes that your existing profile already includes other Systems Manager `ssm` permissions for actions you want to allow access to. This policy alone is not enough to use Session Manager.

To add Session Manager permissions to an existing instance profile (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles.
3. Choose the name of the role to embed a policy in.
4. Choose the Permissions tab.
5. Scroll to the bottom of the page and choose Add inline policy.
6. Choose the JSON tab.
7. Replace the default content with the following:

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel",
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenControlChannel",
                "ssmmessages:OpenDataChannel"
            ],
            "Resource": "*"
        }
    ]
}
```
About 'ssmmessages'

For information about ssmmessages, see Reference: ec2messages, ssmmessages, and other API calls (p. 1195).

About 'kms:Decrypt'

In this policy, the kms:Decrypt permission enables customer key encryption and decryption for session data. If you will use AWS Key Management Service (AWS KMS) encryption for your session data, replace key-name with the ARN of the customer master key (CMK) you want to use, in the format arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE.

If you will not use AWS KMS encryption for your session data, you can remove the following content from the policy:

```
, 
  { 
    "Effect": "Allow",
    "Action": [ 
      "kms:Decrypt"
    ],
    "Resource": "key-name"
  }
```

For information about using AWS KMS and a CMK to encrypt session data, see Enable AWS KMS key encryption of session data (console) (p. 799).

9. On the Review policy page, for Name, enter a name for the inline policy, such as SessionManagerPermissions.
10. Choose Create policy.

Create a custom IAM instance profile for Session Manager

You can create a custom IAM instance profile that provides permissions for only Session Manager actions on your instances. You can also create a policy to provide the permissions needed for logs of session activity to be sent to Amazon S3 and CloudWatch Logs.

After you create an instance profile, see Attaching an IAM Role to an Instance and Attach or Replace an Instance Profile for information about how to attach the instance profile to an instance. For more information about IAM instance profiles and roles, see Using Instance Profile and IAM roles for Amazon EC2 in the IAM User Guide.
Creating an instance profile with minimal Session Manager permissions (console)

Use the following procedure to create a custom IAM instance profile with a policy that provides permissions for only Session Manager actions on your instances.

To create an instance profile with minimal Session Manager permissions (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Policies**, and then choose **Create policy**. (If a **Get Started** button appears, choose it, and then choose **Create Policy**.)
3. Choose the **JSON** tab.
4. Replace the default content with the following:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ssm:UpdateInstanceInformation",
            "ssmmessages:CreateControlChannel",
            "ssmmessages:CreateDataChannel",
            "ssmmessages:OpenControlChannel",
            "ssmmessages:OpenDataChannel"
         ],
         "Resource": "*
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:GetEncryptionConfiguration"
         ],
         "Resource": "*
      },
      {
         "Effect": "Allow",
         "Action": [
            "kms:Decrypt"
         ],
         "Resource": "key-name"
      }
   ]
}
```

**About 'ssmmessages'**

For information about **ssmmessages**, see Reference: ec2messages, ssmmessages, and other API calls (p. 1195).

**About 'kms:Decrypt'**

In this policy, the **kms:Decrypt** permission enables customer key encryption and decryption for session data. If you will use AWS Key Management Service (AWS
KMS) encryption for your session data, replace `key-name` with the ARN of the customer master key (CMK) you want to use, in the format `arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE`.

If you will not use AWS KMS encryption for your session data, you can remove the following content from the policy:

```json
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt"
    ],
    "Resource": "key-name"
}
```

For information about using AWS KMS and a CMK to encrypt session data, see Enable AWS KMS key encryption of session data (console) (p. 799).

5. Choose Review policy.
6. On the Review policy page, for Name, enter a name for the inline policy, such as `SessionManagerPermissions`.
7. (Optional) For Description, enter a description for the policy.
8. Choose Create policy.
9. In the navigation pane, choose Roles, and then choose Create role.
10. On the Create role page, choose AWS service, and from the Choose the service that will use this role list, choose EC2.
11. Choose Next: Permissions.
12. On the Attached permissions policy page, select the check box to the left of name of the policy you just created, such as `SessionManagerPermissions`.
13. Choose Next: Review.
14. On the Review page, for Role name, enter a name for the IAM instance profile, such as `MySessionManagerInstanceProfile`.
15. (Optional) For Role description, enter a description for the instance profile.
16. Choose Create role.

Creating an instance profile with permissions for Session Manager and Amazon S3 and CloudWatch Logs (console)

Use the following procedure to create a custom IAM instance profile with a policy that provides permissions for Session Manager actions on your instances. The policy also provides the permissions needed for session logs to be stored in S3 buckets and CloudWatch Logs log groups.

For information about specifying preferences for storing session logs, see Auditing and logging session activity (p. 823).

To create an instance profile with permissions for Session Manager and Amazon S3 and CloudWatch Logs (console)

1. Sign in to the AWS Management Console and open the IAM console at `https://console.aws.amazon.com/iam/`.
2. In the navigation pane, choose Policies, and then choose Create policy. (If a Get Started button appears, choose it, and then choose Create Policy.)
3. Choose the JSON tab.

4. Replace the default content with the following. Be sure to replace `s3-bucket-name` and `s3-bucket-prefix` with the names for your bucket and its prefix (if any). For information about ssmmessages in the following policy, see Reference: ec2messages, ssmmessages, and other API calls (p. 1195).

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssmmessages:CreateControlChannel",
        "ssmmessages:CreateDataChannel",
        "ssmmessages:OpenControlChannel",
        "ssmmessages:OpenDataChannel",
        "ssm:UpdateInstanceInformation"
      ],
      "Resource": "*
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": "*
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::s3-bucket-name/s3-bucket-prefix"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetEncryptionConfiguration"
      ],
      "Resource": "*
    },
    {
      "Effect": "Allow",
      "Action": "kms:GenerateDataKey",
      "Resource": "*
    }
  ]
}
```

**Important**

To output session logs to an S3 bucket owned by a different AWS account, you must add the IAM `s3:PutObjectAcl` permission to this policy. If this permission isn't added, the account that owns the S3 bucket cannot access the session output logs.

5. Choose Review policy.

6. On the Review policy page, for Name, enter a name for the inline policy, such as `SessionManagerPermissions`.

7. (Optional) For Description, enter a description for the policy.

8. Choose Create policy.
9. In the navigation pane, choose Roles, and then choose Create role.
10. On the Create role page, choose AWS service, and from the Choose the service that will use this role list, choose EC2.
11. Choose Next: Permissions.
12. On the Attached permissions policy page, select the check box to the left of name of the policy you just created, such as SessionManagerPermissions.
13. Choose Next: Review.
14. On the Review page, for Role name, enter a name for the IAM instance profile, such as MySessionManagerInstanceProfile.
15. (Optional) For Role description, enter a description for the instance profile.
16. Choose Create role.

Step 3: Control user session access to instances

Session Manager allows you to centrally grant and revoke user access to instances. Using IAM policies, you control which instances specific users or groups can connect to, and you control what Session Manager API actions they can perform on the instances they are given access to.

About Session ID ARN Formats

IAM policies for Session Manager access use variables for user names as part of session IDs. Session IDs in turn are used in session Amazon Resource Names (ARNs) to control access. Session ARNs have the following format:

```
arn:aws:ssm:region-id:account-id:session/session-id
```

For example:

```
arn:aws:ssm:us-east-2:123456789012:session/JohnDoe-1a2b3c4d5eEXAMPLE
```

You can use a pair of default IAM policies supplied by AWS, one for end users and one for administrators, to supply permissions for Session Manager activities. Or you can create custom IAM policies for different permissions requirements you might have.

For more information about using variables in IAM policies, see IAM Policy Elements: Variables.

For information about how to create policies and attach them to IAM users or groups, see Creating IAM Policies and Adding and Removing IAM Policies in the IAM User Guide.

Topics

- Enforce a session document permission check for the AWS CLI (p. 782)
- Quickstart default IAM policies for Session Manager (p. 784)
- Additional sample IAM policies for Session Manager (p. 790)

Enforce a session document permission check for the AWS CLI

When you configure Session Manager for your account, the system creates a Session-type SSM document named SSM-SessionManagerRunShell. This SSM document stores your session preferences, such as whether session data is saved in an S3 bucket or Amazon CloudWatch Logs log group, whether session data is encrypted using AWS Key Management Service, and whether Run As support is enabled for your sessions. The following is an example.
By default, if a user in your account was granted permission in their IAM user policy to start sessions, that user has access to the SSM-SessionManagerRunShell SSM document. This means that when they use the AWS CLI to run the `start-session` command, and they do not specify a document in the `--document-name` option, the system uses SSM-SessionManagerRunShell and launches the session. The session starts even if the user's IAM policy doesn't grant explicit permission to access the SSM-SessionManagerRunShell document.

For example, the following command doesn't specify a session document.

```bash
aws ssm start-session \
  --target i-02573cafcfEXAMPLE
```

The following example specifies the default Session Manager session document.

```bash
aws ssm start-session \
  --document-name SSM-SessionManagerRunShell \
  --target i-02573cafcfEXAMPLE
```

To restrict access to the default or any session document, you can add a condition element to the user's IAM policy that validates whether the user has explicit access to a session document. When this condition is applied, the user must specify a value for the `--document-name` option of the `start-session` AWS CLI command. This value is either the default Session Manager session document or a custom session document you created. The following condition element, when added to the `ssm:StartSession` action in the IAM policy, performs a session document access check.

```json
"Condition": {
    "BoolIfExists": {
        "ssm:SessionDocumentAccessCheck": "true"
    }
}
```

With this condition element set to `true`, explicit access to a session document must be granted in the IAM policy for the user to start a session. The following is an example.

```json
{
    "Effect": "Allow",
    "Action": [
        "ssm:StartSession"
    ],
    "Resource": [
        "arn:aws:ec2:*:*:instance/instance-id",
        "arn:aws:ssm:region:account-id:document/SSM-SessionManagerRunShell"
    ]
}
```
For an example of specifying a Session Manager session document in an IAM policy, see Quickstart end user policies for Session Manager (p. 784).

Other Scenarios

Using the default SSM-SessionManagerRunShell session document is the only case when a document name can be omitted from the `start-session` CLI command. In other cases, the user must specify a value for the `--document-name` option of the `start-session` AWS CLI command. The system checks whether the user has explicit access to the session document they specify.

For example, if a user specifies the name of a custom session document you created, the user’s IAM policy must grant them permission to access that document.

If a user runs a command to start a session using SSH, the user’s policy must grant them access to the AWS-StartSSHSession session document.

**Note**

To start a session using SSH, configuration steps must be completed on both the target instance and the user's local machine. For information, see (Optional) Enable SSH connections through Session Manager (p. 810).

Quickstart default IAM policies for Session Manager

Use the samples in this section to help you create IAM policies that provide the most commonly needed permissions for Session Manager access.

**Note**

You can also use an AWS KMS key policy to control which IAM users, IAM roles, and AWS accounts are given access to your CMK. For information, see Overview of Managing Access to Your AWS KMS Resources and Using Key Policies in AWS KMS in the AWS Key Management Service Developer Guide.

Topics

- Quickstart end user policies for Session Manager (p. 784)
- Quickstart administrator policy for Session Manager (p. 787)

Quickstart end user policies for Session Manager

Use the following examples to create IAM end user policies for Session Manager.

You can create a policy that allows users to start sessions from only the Session Manager console and AWS CLI, from only the Amazon EC2 console, or from all three.

These policies provide end users the ability start a session to a particular instance and the ability to end only their own sessions. Refer to Additional sample IAM policies for Session Manager (p. 790) for examples of customizations you might want to make to the policy.

**Note**

In all the following sample policies, replace `instance-id` with the ID of the instance you want to grant access to, in the format `i-02573cafcfEXAMPLE`. Replace `region` and `account-id` with your AWS Region and AWS Account ID, such as `us-east-2` and `111122223333`.

Refer to the following sections to view sample policies for the range of session access you want to provide.
Session Manager and CLI

Use this sample policy to provide users with the ability to start sessions from only the Session Manager console and the AWS CLI. This policy doesn't provide all the permissions needed to start sessions from the Amazon EC2 console.

```json
{
   "Version": "2012-10-17",
   "Statement": [
    {
       "Effect": "Allow",
       "Action": ["ssm:StartSession"],
       "Condition": {
          "BoolIfExists": {
             "ssm:SessionDocumentAccessCheck": "true"
          }
       }
    },
    {
       "Effect": "Allow",
       "Action": ["ssm:DescribeSessions", "ssm:GetConnectionStatus", "ssm:DescribeInstanceProperties", "ec2:DescribeInstances"],
       "Resource": "*"
    },
    {
       "Effect": "Allow",
       "Action": ["ssm:TerminateSession"],
       "Resource": ["arn:aws:ssm:*:*:session/${aws:username}*"],
    },
    {
       "Effect": "Allow",
       "Action": ["kms:GenerateDataKey"],
       "Resource": "key-name"
    }
   ]
}
```

Amazon EC2

Use this sample policy to provide users with the ability to start sessions from only the Amazon EC2 console. This policy doesn't provide all the permissions needed to start sessions from the Session Manager console and the AWS CLI.

```json
{
   "Version": "2012-10-17",
   "Statement": ["arn:aws:ssm:*:*:document/SSM-SessionManagerRunShell"
```
Session Manager, CLI, and Amazon EC2

Use this sample policy to provider users with the ability to start sessions from the Session Manager console, the AWS CLI, and the Amazon EC2 console.
Getting started with Session Manager

1 SSM-SessionManagerRunShell is the default name of the SSM document that Session Manager creates to store your session configuration preferences. You can create a custom session document and specify it in this policy instead. You can also specify the AWS-provided document, AWS-StartSSHSession, for users who are starting sessions using SSH. For information about configuration steps needed to support sessions using SSH, see (Optional) Enable SSH connections through Session Manager (p. 810).

2 If you specify the condition element, ssm:SessionDocumentAccessCheck, as true, the system checks that a user has explicit access to the defined session document, in this example SSM-SessionManagerRunShell, before a session is established. For more information, see Enforce a session document permission check for the AWS CLI (p. 782).

3 The kms:GenerateDataKey permission enables the creation of a data encryption key that will be used to encrypt session data. If you will use AWS Key Management Service (AWS KMS) encryption for your session data, replace key-name with the ARN of the customer master key (CMK) you want to use, in the format arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE. If you won’t use AWS KMS key encryption for your session data, remove the following content from the policy:

```json
],
  "Effect": "Allow",
  "Action": [
    "kms:GenerateDataKey"
  ],
  "Resource": "key-name"
}
```

For information about AWS KMS and CMKs for encrypting session data, see Enable AWS KMS key encryption of session data (console) (p. 799).

4 The permission for SendCommand is needed for cases where a user attempts to start a session from the Amazon EC2 console, but a command must be sent to update SSM Agent first.

Quickstart administrator policy for Session Manager

Use the following examples to create IAM administrator policies for Session Manager.
These policies provide administrators the ability to start a session to instances that are tagged with Key=Finance, Value=WebServers, permission to create, update and delete preferences, and permission to end only their own sessions. Refer to Additional sample IAM policies for Session Manager (p. 790) for examples of customizations you might want to make to the policy.

You can create a policy that allows administrators to perform these tasks from only the Session Manager console and AWS CLI, from only the Amazon EC2 console, or from all three.

**Note**
Replace `region` and `account-id` with your AWS Region and AWS Account ID, such as us-east-2 and 111122223333.

Refer to the following sections to view sample policies for the three permissions scenarios.

**Session Manager and CLI**

Use this sample policy to provide administrators with the ability to perform session-related tasks from only the Session Manager console and the AWS CLI. This policy doesn't provide all the permissions needed to perform session-related tasks from the Amazon EC2 console.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:StartSession"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*"
            ],
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/Finance": [
                        "WebServers"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeSessions",
                "ssm:GetConnectionStatus",
                "ssm:DescribeInstanceProperties",
                "ec2:DescribeInstances"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:CreateDocument",
                "ssm:UpdateDocument",
                "ssm:GetDocument"
            ],
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:TerminateSession"
            ]
        }
    ]
}
```
Use this sample policy to provide administrators with the ability to perform session-related tasks from only the Amazon EC2. This policy doesn't provide all the permissions needed to perform session-related tasks from the Session Manager console and the AWS CLI.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ssm:StartSession",
            "ssm:SendCommand"
         ],
         "Resource": [
            "arn:aws:ec2:*:*:instance/*"
         ],
         "Condition": {
            "StringLike": {
               "ssm:resourceTag/tag-key": [
                  "tag-value"
               ]
            }
         }
      },
      {
         "Effect": "Allow",
         "Action": [
            "ssm:GetConnectionStatus",
            "ssm:DescribeInstanceInformation"
         ],
         "Resource": "*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "ssm:TerminateSession"
         ],
         "Resource": [
            "arn:aws:ssm:*:*:session/${aws:username}-*"
         ]
      }
   ]
}
```
The permission for `SendCommand` is needed for cases where a user attempts to start a session from the Amazon EC2 console, but a command must be sent to update SSM Agent first.

**Additional sample IAM policies for Session Manager**

Refer to the following example policies to help you create a custom IAM policy for any Session Manager user access scenarios you want to support.

**Topics**
- Example 1: Restrict access to specific instances (p. 791)
- Example 2: Restrict access based on instance tags (p. 791)
- Example 3: Allow a user to end only sessions they started (p. 792)
Example 1: Restrict access to specific instances

You can restrict access to specific instances by creating an IAM user policy that includes the IDs of the instances. In the following example, the user is allowed Session Manager access to three specific instances only, and allowed to end only their sessions on those instances. If the user sends a command to any other instance or tries to end any other session, the command result will include AccessDenied.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ssm:StartSession"],
      "Resource": [
        "arn:aws:ec2:us-east-2:123456789012:instance/i-0e9d8c7b6aEXAMPLE"
      ]
    },
    {
      "Effect": "Allow",
      "Action": ["ssm:TerminateSession"],
      "Resource": ["arn:aws:ssm:*::*:session/${aws:username}-*"
      ]
    }
  ]
}
```

Example 2: Restrict access based on instance tags

You can restrict access to instances based on specific Amazon EC2 tags. In the following example, the user is allowed to start sessions (Effect: Allow, Action: ssm:StartSession) on any instance (Resource: arn:aws:ec2:*::*:instance/*) with the condition that the instance is a Finance WebServer (ssm:resourceTag/Finance: WebServer). If the user sends a command to an instance that is not tagged or that has any tag other than Finance: WebServer, the command result will include AccessDenied.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ssm:StartSession"],
      "Resource": ["arn:aws:ec2:*::*:instance/*"]
    },
    "Condition": {
      "StringLike": {
        "ssm:resourceTag/Finance": ["WebServers"
      ]
    }
  ]
}
```
You can create IAM policies that enable a user to start sessions to instances that are tagged with multiple tags. The following policy enables the user to start sessions to instances that have both the specified tags applied to them. If a user sends a command to an instance that is not tagged with both of these tags, the command result will include AccessDenied.

```json
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Effect": "Allow",
            "Action": [ "ssm:StartSession"
            ],
            "Resource": [ "*" ],
            "Condition":{
                "StringLike":{
                    "ssm:resourceTag/tag-key1": [ "tag-value1" ],
                    "ssm:resourceTag/tag-key2": [ "tag-value2"
                }
            }
        }
    ]
}
```

For more information about creating IAM user policies, see Managed Policies and Inline Policies in the IAM User Guide. For more information about tagging instances, see Tagging your Amazon EC2 resources in the Amazon EC2 User Guide for Linux Instances (content applies to Windows and Linux instances). For more information increasing your security posture against unauthorized root-level commands on your instances, see Restrict access to root-level commands through SSM Agent (p. 92)

**Example 3: Allow a user to end only sessions they started**

Session Manager provides two methods to control which sessions a user in your AWS account is allowed to end.

- **Use the variable {aws:username} in an AWS Identity and Access Management (IAM) permissions policy.** Users can end only sessions they started. This method does not work for accounts that use federated IDs to grant access to AWS. Federated IDs use the variable {aws:userid} instead of {aws:username}.
- **Use tags supplied by AWS tags in an IAM permissions policy.** In the policy, you include a condition that allows users to end only sessions that are tagged with specific tags that have been provided by AWS. This method works for all accounts, including those that use federated IDs to grant access to AWS.
Method 1: Grant TerminateSession privileges using the variable {aws:username}

The following IAM policy lets a user view the IDs of all sessions in your account. However, users can interact with instances only through sessions they started. A user who is assigned the following policy can't connect to or end other users' sessions. The policy uses the variable {aws:username} to achieve this.

**Note**
This method does not work for accounts that grant access to AWS using federated IDs.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ssm:DescribeSessions"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    },
    {
      "Action": [
        "ssm:TerminateSession"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:ssm:*:*:session/${aws:username}-*
      ]
    }
  ]
}
```

Method 2: Grant TerminateSession privileges using tags supplied by AWS

You can control which sessions a user can end by using a condition with specific tag key variables in an IAM user policy. The condition specifies that the user can only end sessions that are tagged with one or both of these specific tag key variables and a specified value.

When a user in your AWS account starts a session, Session Manager applies two resource tags to the session. The first resource tag is `aws:ssmmessages:target-id`, with which you specify the ID of the target the user is allowed to end. The other resource tag is `aws:ssmmessages:session-id`, with a value in the format of `role-id:caller-specified-role-name`.

**Note**
Session Manager doesn't support custom tags for this IAM access control policy. You must use the resource tags supplied by AWS, described below.

**aws:ssmmessages:target-id**

With this tag key, you include the instance ID as the value in policy. In the following policy block, the condition statement lets a user end only the instance i-02573cafcf

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:TerminateSession"
      ],
      "Resource": "",
    }
  ]
}
```
If the user tries to end a session for which they haven’t been granted this \texttt{TerminateSession} permission, they receive an \texttt{AccessDeniedException} error.

**aws:ssmmessages:session-id**

This tag key includes a variable for the session ID as the value in the request to start a session.

The following example demonstrates a policy for cases where the caller type is User. The value you supply for \texttt{aws:ssmmessages:session-id} is the ID of the user. In this example, \texttt{AIDIODR4TAW7CSEXAMPLE} represents the ID of a user in your AWS account. To retrieve the ID for a user in your AWS account, use the IAM command, `get-user`. For information, see `get-user` in the AWS Identity and Access Management section of the IAM User Guide.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ssm:TerminateSession"],
            "Resource": "",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": ["AIDIODR4TAW7CSEXAMPLE"]
                }
            }
        }
    ]
}
```

The following example demonstrates a policy for cases where the caller type is \texttt{AssumedRole}. You can use the \texttt{\{aws:userid\}} variable for the value you supply for \texttt{aws:ssmmessages:session-id}. Alternatively, you can hardcode a role ID for the value you supply for \texttt{aws:ssmmessages:session-id}. If you hardcode a role ID, you must provide the value in the format \texttt{role-id:caller-specified-role-name}. For example, \texttt{AIDIODR4TAW7CSEXAMPLE:MyRole}.

**Important**

In order for system tags to be applied, the role ID you supply can contain the following characters only: Unicode letters, 0-9, space, \_\, \., /, =, +, - and \.`.

To retrieve the role ID for a role in your AWS account, use the `get-caller-identity` command. For information, see `get-caller-identity` in the AWS CLI Command Reference.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ssm:TerminateSession"],
            "Resource": "",
            "Condition": {
                "StringLike": {
                    "ssm:resourceTag/aws:ssmmessages:session-id": ["AIDIODR4TAW7CSEXAMPLE:MyRole"]
                }
            }
        }
    ]
}
```
If a user tries to end a session for which they haven't been granted this `TerminateSession` permission, they receive an `AccessDeniedException` error.

`aws:ssmmessages:target-id` and `aws:ssmmessages:session-id`

You can also create IAM policies that enable a user to end sessions that are tagged with both system tags, as shown in this example.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:TerminateSession"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/aws:ssmmessages:target-id": [
            "instance-id"
          ],
          "ssm:resourceTag/aws:ssmmessages:session-id": [
            "${aws:username}-*"
          ]
        }
      }
    }
  ]
}
```

Example 4: Allow full (administrative) access to all sessions

The following IAM policy allows a user to fully interact with all instances and all sessions created by all users for all instances. It should be granted only to an Administrator who needs full control over your organization's Session Manager activities.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ssm:StartSession",
        "ssm:TerminateSession",
        "ssm:ResumeSession",
        "ssm:DescribeSessions",
        "ssm:GetConnectionStatus"
      ]
    }
  ]
}
```
Step 4: Configure session preferences

An IAM user with administrator permissions can do the following:

- Enable Run As support for Linux instances. This makes it possible to start sessions using the credentials of a specified operating system user instead of the credentials of a system-generated `ssm-user` account that Session Manager can create on a managed instance.
- Configure Session Manager to use AWS KMS key encryption to provide additional protection to the data transmitted between client machines and managed instances.
- Configure Session Manager to create and send session history logs to an Amazon Simple Storage Service (Amazon S3) bucket or an Amazon CloudWatch Logs log group. The stored log data can then be used to audit or report on the session connections made to your instances and the commands run on them during the sessions.

**Note**

Before a user can update Session Manager preferences, they must have been granted the specific permissions that will let them make these updates, if they do not possess them already. Without these permissions, the user can’t configure logging options or set other session preferences for your account.

Topics

- Grant or deny a user permissions to update Session Manager preferences (p. 796)
- Enable run as support for Linux instances (p. 797)
- Enable AWS KMS key encryption of session data (console) (p. 799)
- Create Session Manager preferences (command line) (p. 800)
- Update Session Manager preferences (command line) (p. 802)

For information about using the Systems Manager console to configure options for logging session data, see the following topics.

- Logging session data using Amazon S3 (console) (p. 824)
- Logging session data using Amazon CloudWatch Logs (console) (p. 825)

Grant or deny a user permissions to update Session Manager preferences

Account preferences are stored as SSM documents for each AWS Region. Before a user can update account preferences for sessions in your account, they must be granted the necessary permissions to access the type of SSM document where these preferences are stored. These permissions are granted through an IAM policy.

**Administrator policy to allow preferences to be created and updated**

An administrator can have the following policy to create and update preferences at any time. The following policy allows permission to access and update the `SSM-SessionManagerRunShell` document in the us-east-2 account 123456789012.
User policy to prevent preferences from being updated

Use the following policy to prevent end users in your account from updating or overriding any Session Manager preferences.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ssm:CreateDocument",
        "ssm:UpdateDocument",
        "ssm:DeleteDocument"
      ],
      "Effect": "Deny",
      "Resource": [
      ]
    }
  ]
}
```

Enable run as support for Linux instances

By default, sessions are launched using the credentials of a system-generated `ssm-user` account that is created on a managed instance. (On Linux machines, this account is added to `/etc/sudoers/`.) You can instead launch sessions using the credentials of an operating system account. Session Manager provides two methods for specifying the operating system account to use.

**Method 1: Tag an IAM user or role (recommended)**

You can specify the operating system user account that is used to start sessions by tagging an IAM user or associated role with the AWS-provided key name `SSMSessionRunAs`, and specifying the OS user name as its value. For example, if the OS user account name is `DevRoleLogin`, the corresponding tag to use is `SSMSessionRunAs = DevRoleLogin`.

Using this method, you could specify a different OS account name for each IAM user or role you tag, or use the same OS user name for them all.

For more information about tagging IAM entities, see the following topics:

- [Tagging IAM Entities](#) in the [IAM User Guide](#)
- [Add Tags to Manage Your AWS IAM Users and Roles](#) on the [AWS Security Blog](#)
Method 2: Specify an OS user name in Session Manager preferences

When you configure Session Manager preferences in the console or by using the AWS CLI, you can specify the operating system user name to start sessions with.

Using this method, all sessions are run by the same OS user for all the IAM users in your account who connect to the instance using Session Manager.

How It Works

If you enable Run As support for sessions, the system checks for access permissions as follows:

1. For the user who is starting the session, has their IAM user account or role been tagged with `SSMSessionRunAs = os-user-account-name`?
   
   If Yes, does the user name exist on the instance? If it does, start the session. If it does not, do not allow a session to start.
   
   If the IAM user's account or role has not been tagged with `SSMSessionRunAs = os-user-account-name`, continue to step 2.

2. If the IAM user's account or role hasn't been tagged with `SSMSessionRunAs = os-user-account-name`, has an OS user name been specified in the AWS account's Session Manager preferences?
   
   If Yes, does the user name exist on the instance? If it does, start the session. If it does not, do not allow a session to start.

At this point, Session Manager does not fall back on the default `ssm-user` account. In other words, enabling Run As support prevents sessions from being started using an `ssm-user` account on an instance.

To enable run as support for Linux instances

2. In the navigation pane, choose Session Manager.
3. Choose the Preferences tab, and then choose Edit.
4. Select the check box next to Enable Run As support for Linux instances.
5. Do one of the following:
   
   • **Option 1**: For (Optional) Enter an operating system user name for starting sessions, enter the name of the operating system user account on the target instance that you want to use to start sessions.
   
   • **Option 2**: Choose the IAM console link. In the navigation pane, choose either Users or Roles. Choose the entity (user or role) to add tags to, and then choose the Tags tab. Enter `SSMSessionRunAs` for the key name. Enter the name of a user account on your target instance for the key value. Choose Save changes.

   The following is an example.
Enable AWS KMS key encryption of session data (console)

Use AWS Key Management Service (AWS KMS) to create and manage keys. With AWS KMS, you can control the use of encryption across a wide range of AWS services and in your applications. You can specify that session data transmitted between your EC2 instances and the local machines of users in your AWS account is encrypted using AWS KMS key encryption. (This is in addition to the TLS 1.2 encryption that AWS already provides by default.) AWS KMS key encryption for sessions is accomplished using a customer master key (CMK) that is created in AWS KMS.

Note
You must enable AWS KMS encryption in order to reset passwords on your managed instances from the Systems Manager console. For more information, see Reset a password on a managed instance (p. 766).

You can use a key that you created in your AWS account. You can also use a key that was created in a different AWS account. The creator of the key in a different AWS account must provide you with the permissions needed to use the key.

After you enable AWS KMS key encryption for your session data, both the users who start sessions and the instances that they connect to must have permission to use the key. You provide permission to use the CMK with Session Manager through IAM policies. For information, see the following topics:

- Add CMK permissions for users in your account: Quickstart default IAM policies for Session Manager (p. 784).
- Add CMK permissions for instances in your account: Step 2: Verify or create an IAM instance profile with Session Manager permissions (p. 776).

For more information about creating and managing AWS KMS keys, see the AWS Key Management Service Developer Guide.

For information about using the AWS CLI to enable AWS KMS key encryption of session data in your account, see Create Session Manager preferences (command line) (p. 800) or Update Session Manager preferences (command line) (p. 802).

Note
There is a charge to use CMKs. For information, see AWS Key Management Service pricing.

To enable AWS KMS key encryption of session data (console)

2. In the navigation pane, choose Session Manager.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Select the check box next to **Key Management Service (KMS)**.
5. Do one of the following:
   - Choose the button next to **Select an AWS KMS key in my current account**, then select a key from the list.
   - Choose the button next to **Enter a KMS key alias or KMS key ARN**. Manually enter an AWS KMS key alias for a key created in your current account, or enter the key ARN for a key in another account. The following are examples.
     - Key alias: `alias/my-kms-key-alias`
     - Key ARN: `arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-12345EXAMPLE`
   - Choose **Create new key** to create a new CMK in your account. After you create the new key, return to the **Preferences** tab and select the key for encrypting session data in your account.

   For more information about sharing keys, see **Allowing External AWS Accounts to Access a CMK** in the **AWS Key Management Service Developer Guide**.
6. Choose **Save**.

### Create Session Manager preferences (command line)

The following procedure describes how to use your preferred command line tool to create Session Manager preferences for your account in the selected AWS Region. Use Session Manager preferences to specify options for logging session data in an S3 bucket or Amazon CloudWatch Logs log group. You can also use Session Manager preferences to encrypt your session data.

For information about using command line tools to update existing Session Manager preferences, see **Update Session Manager preferences (command line)** (p. 802).

#### To create Session Manager preferences (command line)

1. Create a JSON file on your local machine with a name such as **SessionManagerRunShell.json**, and then paste the following content into it:

   ```json
   {
   "schemaVersion": "1.0",
   "description": "Document to hold regional settings for Session Manager",
   "sessionType": "Standard_Stream",
   "inputs": {
       "s3BucketName": "",
       "s3KeyPrefix": "",
       "s3EncryptionEnabled": true,
       "cloudWatchLogGroupName": "",
       "cloudWatchEncryptionEnabled": true,
       "kmsKeyId": "",
       "runAsEnabled": "",
       "runAsDefaultUser": ""
   }
   }
   
   2. Specify where you want to send session data. You can specify an S3 bucket name (with an optional prefix) or a CloudWatch Logs log group name. If you want to further encrypt data between local
client and EC2 instances, provide the AWS KMS key to use for encryption. The following is an example.

```json
{
   "schemaVersion": "1.0",
   "description": "Document to hold regional settings for Session Manager",
   "sessionType": "Standard_Stream",
   "inputs": {
      "s3BucketName": "MyBucketName",
      "s3KeyPrefix": "MyBucketPrefix",
      "s3EncryptionEnabled": true,
      "cloudWatchLogGroupName": "MyLogGroupName",
      "cloudWatchEncryptionEnabled": true,
      "kmsKeyId": "MyKMSKeyID",
      "runAsEnabled": true,
      "runAsDefaultUser": "MyDefaultRunAsUser"
   }
}
```

**Note**

If you do not want to encrypt the session log data, change "true" to "false" for `s3EncryptionEnabled`. If you aren't sending logs to either an S3 bucket or a CloudWatch Logs log group, don't want to encrypt active session data, or don't want to enable Run As support for the sessions in your account, you can delete the lines for those options. Make sure the last line in the "inputs" section does not end with a comma.

If you add a AWS KMS key ID to encrypt your session data, both the users who start sessions and the instances that they connect to must have permission to use the key. You provide permission to use the CMK with Session Manager through IAM policies. For information, see the following topics:

- Add CMK permissions for users in your account: Quickstart default IAM policies for Session Manager (p. 784).
- Add CMK permissions for instances in your account: Step 2: Verify or create an IAM instance profile with Session Manager permissions (p. 776).

3. Save the file.
4. In the directory where you created the JSON file, run the following command:

   **Linux**
   ```bash
   aws ssm create-document \
   --name SSM-SessionManagerRunShell \
   --content "file://SessionManagerRunShell.json" \
   --document-type "Session" \
   --document-format JSON
   ```

   **Windows**
   ```bash
   aws ssm create-document ^
   --name SSM-SessionManagerRunShell ^
   --content "file://SessionManagerRunShell.json" ^
   --document-type "Session" ^
   --document-format JSON
   ```

   **PowerShell**
   ```bash
   New-SSMDocument `
   -Name "SSM-SessionManagerRunShell" `
   ```
Update Session Manager preferences (command line)

The following procedure describes how to use your preferred command line tool to make changes to the Session Manager preferences for your account in the selected AWS Region. Use Session Manager preferences to specify options for logging session data in an S3 bucket or Amazon CloudWatch Logs log group. You can also use Session Manager preferences to encrypt your session data.

To update Session Manager preferences (command line)

1. Create a JSON file on your local machine with a name such as `SessionManagerRunShell.json`, and then paste the following content into it:

```json
{
    "schemaVersion": "1.0",
    "description": "Document to hold regional settings for Session Manager",
    "sessionType": "Standard_Stream",
    "inputs": {
        "s3BucketName": "",
        "s3KeyPrefix": "",
        "s3EncryptionEnabled": true,
        "cloudWatchLogGroupName": "",
        "cloudWatchEncryptionEnabled": true,
        "kmsKeyId": "",
        "runAsEnabled": true,
        "runAsDefaultUser": ""
    }
}
```

2. Specify where you want to send session data. You can specify an S3 bucket name (with an optional prefix) or a CloudWatch Logs log group name. If you want to further encrypt data between local...
client and EC2 instances, provide the AWS KMS key to use for encryption. The following is an example.

```json
{
    "schemaVersion": "1.0",
    "description": "Document to hold regional settings for Session Manager",
    "sessionType": "Standard_Stream",
    "inputs": {
        "s3BucketName": "MyBucketName",
        "s3KeyPrefix": "MyBucketPrefix",
        "s3EncryptionEnabled": true,
        "cloudWatchLogGroupName": "MyLogGroupName",
        "cloudWatchEncryptionEnabled": true,
        "kmsKeyId": "MyKMSKeyID",
        "runAsEnabled": true,
        "runAsDefaultUser": "MyDefaultRunAsUser"
    }
}
```

**Note**

If you do not want to encrypt the session log data, change "true" to "false" for `s3EncryptionEnabled`.

If you aren't sending logs to either an S3 bucket or a CloudWatch Logs log group, don't want to encrypt active session data, or don't want to enable Run As support for the sessions in your account, you can delete the lines for those options. Make sure the last line in the "inputs" section does not end with a comma.

If you add a AWS KMS key ID to encrypt your session data, both the users who start sessions and the instances that they connect to must have permission to use the key. You provide permission to use the CMK with Session Manager through IAM policies. For information, see the following topics:

- Add CMK permissions for users in your account: Quickstart default IAM policies for Session Manager (p. 784).
- Add CMK permissions for instances in your account: Step 2: Verify or create an IAM instance profile with Session Manager permissions (p. 776).

3. Save the file.
4. In the directory where you created the JSON file, run the following command:

**Linux**

```bash
aws ssm update-document \ 
    --name "SSM-SessionManagerRunShell" \ 
    --content "file://SessionManagerRunShell.json" \ 
    --document-version "$LATEST"
```

**Windows**

```bash
aws ssm update-document ^ 
    --name "SSM-SessionManagerRunShell" ^ 
    --content "file://SessionManagerRunShell.json" ^ 
    --document-version "$LATEST"
```

**PowerShell**

```powershell
Update-SSMDocument ` 
-Name "SSM-SessionManagerRunShell" ` 
-Content (Get-Content -Raw SessionManagerRunShell.json) ` 
```
If successful, the command returns output similar to the following:

```json
{
   "DocumentDescription": {
      "Status": "Updating",
      "Hash": "ce4f4d0a2ab9b0f8e759004ba603174c3ec2231f21a81db8690a33eb66EXAMPLE",
      "Name": "SSM-SessionManagerRunShell",
      "Tags": [],
      "DocumentType": "Session",
      "PlatformTypes": [
         "Windows",
         "Linux"
      ],
      "DocumentVersion": "2",
      "HashType": "Sha256",
      "CreatedDate": 1537206341.565,
      "Owner": "111122223333",
      "SchemaVersion": "1.0",
      "DefaultVersion": "1",
      "DocumentFormat": "JSON",
      "LatestVersion": "2"
   }
}
```

### Step 5: (Optional) Restrict access to commands in a session

You can restrict the commands a user can run in a Session Manager session by creating a custom Session type SSM document. In the document content, you define which command is run when the user starts a session and what parameters they can provide to the command. These are also referred to as interactive commands. The Session document schemaVersion must be 1.0 and the sessionType of the document must be InteractiveCommands. You can then create IAM policies that allow users to access only the Session documents you define. For more information about using IAM policies to restrict access to commands in a session, see [IAM policy examples for interactive commands](p. 806).

The user specifies the allowed document in the --document-name option for the start-session command and provides any necessary parameter values for the command in the --parameters option. For more information about running interactive commands, see [Starting a session (interactive commands)](p. 821).

The following procedure describes how to create a custom Session type SSM document that defines the command a user is allowed to run.

#### Restrict access to commands in a session (console)

**To restrict the commands a user can run in a Session Manager session (console)**

2. In the navigation pane, choose Documents.
3. Choose **Create command or session**.
4. For **Name**, type a descriptive name for the document.
5. For **Document type**, choose **Session document**.
6. Enter your document content that defines the command a user can run in a Session Manager session using JSON or YAML, as shown in the following example.
7. Choose Create document.

Restrict access to commands in a session (command line)

Before you begin

Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already. For information, see Install or upgrade AWS command line tools (p. 57).

To restrict the commands a user can run in a Session Manager session (command line)

1. Create a JSON or YAML file for your document content that defines the command a user can run in a Session Manager session, as shown in the following example.

YAML

```yaml
---
schemaVersion: '1.0'
description: Document to view a log file on a Linux instance
sessionType: InteractiveCommands
parameters:
  logpath:
    type: String
description: The log file path to read.
default: "/var/log/amazon/ssm/amazon-ssm-agent.log"
properties:
  linux:
    commands: "tail -f {{ logpath }}"
    runAsElevated: true
```

JSON

```json
{
  "schemaVersion": "1.0",
  "description": "Document to view a log file on a Linux instance",
  "sessionType": "InteractiveCommands",
  "parameters": {
    "logpath": {
      "type": "String",
      "description": "The log file path to read.",
      "default": "/var/log/amazon/ssm/amazon-ssm-agent.log"
    }
  },
  "properties": {
    "linux": {
      "commands": "tail -f {{ logpath }}",
      "runAsElevated": true
    }
  }
}
```
2. Run the following commands to create an SSM document using your content that defines the command a user can run in a Session Manager session.

**Linux**

```
aws ssm create-document \
--content file://path/to/file/documentContent.json \
--name "exampleAllowedSessionDocument" \
--document-type "Session"
```

**Windows**

```
aws ssm create-document ^
--content file://C:\path\to\file\documentContent.json ^
--name "exampleAllowedSessionDocument" ^
--document-type "Session"
```

**PowerShell**

```
$json = Get-Content -Path "C:\path\to\file\documentContent.json" | Out-String
New-SSMDocument -Content $json \
-Name "exampleAllowedSessionDocument" \
-DocumentType "Session"
```

### IAM policy examples for interactive commands

You can create IAM policies that allow users to access only the `Session` documents you define. This restricts the commands a user can run in a Session Manager session to only the commands defined in your custom `Session` type SSM documents.
Allow a user to run an interactive command on a single instance

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":"ssm:StartSession",
      "Resource":[
      ],
      "Condition":{
        "BoolIfExists":{
          "ssm:SessionDocumentAccessCheck":"true"
        }
      }
    }
  ]
}
```

Allow a user to run an interactive command on all instances

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":"ssm:StartSession",
      "Resource":[
      ],
      "Condition":{
        "BoolIfExists":{
          "ssm:SessionDocumentAccessCheck":"true"
        }
      }
    }
  ]
}
```

Allow a user to run multiple interactive commands on all instances

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":"ssm:StartSession",
      "Resource":[
      ],
      "Condition":{
        "BoolIfExists":{
          "ssm:SessionDocumentAccessCheck":"true"
        }
      }
    }
  ]
}
```
Step 6: (Optional) use PrivateLink to set up a VPC endpoint for Session Manager

You can further improve the security posture of your managed instances by configuring AWS Systems Manager to use an interface VPC endpoint. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access Amazon EC2 and Systems Manager APIs by using private IP addresses.

PrivateLink restricts all network traffic between your managed instances, Systems Manager, and Amazon EC2 to the Amazon network. (Managed instances don't have access to the internet.) Also, you don't need an internet gateway, a NAT device, or a virtual private gateway.

In addition to the three endpoints required to use PrivateLink with Systems Manager, you can create a fourth, `com.amazonaws.region.ssmmessages`, for use with Session Manager.

For more information, see (Optional) Create a Virtual Private Cloud endpoint (p. 37).

Step 7: (Optional) disable or enable ssm-user account administrative permissions

Starting with version 2.3.50.0 of SSM Agent, the agent creates a local user account called `ssm-user` and adds it to `/etc/sudoers` (Linux) or to the Administrators group (Windows). On agent versions before 2.3.612.0, the account is created the first time SSM Agent starts or restarts after installation. On version 2.3.612.0 and later, the `ssm-user` account is created the first time a session is started on an instance. This `ssm-user` is the default OS user when a Session Manager session is started.

If you want to prevent Session Manager users from running administrative commands on an instance, you can update the `ssm-user` account permissions. You can also restore these permissions after they have been removed.

Topics

- Managing ssm-user sudo account permissions on Linux (p. 808)
- Managing ssm-user Administrator account permissions on Windows server (p. 809)

Managing ssm-user sudo account permissions on Linux

Use one of the following procedures to disable or enable the ssm-user account sudo permissions on Linux instances:

Use Run Command to modify ssm-user sudo permissions (console)

- Use the procedure in Running commands from the console (p. 834) with the following values:
  - For Command document, choose AWS-RunShellScript.
  - To remove sudo access, in the Command parameters area, paste the following in the Commands box:

```
  cd /etc/sudoers.d
  echo "#User rules for ssm-user" > ssm-agent-users
```
Use the command line to modify ssm-user sudo permissions (AWS CLI)

1. Connect to the instance and run the following command:

   ```
   sudo -s
   ```

2. Change the working directory using the following command:

   ```
   cd /etc/sudoers.d
   ```

3. Open the file named `ssm-agent-users` for editing.

4. To remove sudo access, delete the following line:

   ```
   ssm-user ALL=(ALL) NOPASSWD:ALL
   ```

   -or-

   To restore sudo access, add the following line:

   ```
   ssm-user ALL=(ALL) NOPASSWD:ALL
   ```

5. Save the file.

Managing ssm-user Administrator account permissions on Windows server

Use one of the following procedures to disable or enable the ssm-user account Administrator permissions on Windows Server instances:

Use Run Command to modify Administrator permissions (console)

- Use the procedure in Running commands from the console (p. 834) with the following values:

  For **Command document**, choose AWS-RunPowerShellScript.

  To remove administrative access, in the **Command parameters** area, paste the following in the **Commands** box:

  ```
  net localgroup "Administrators" "ssm-user" /delete
  ```

  -or-

  To restore administrative access, in the **Command parameters** area, paste the following in the **Commands** box:

  ```
  net localgroup "Administrators" "ssm-user" /add
  ```
Use the PowerShell or command prompt window to modify Administrator permissions

1. Connect to the instance and open the PowerShell or Command Prompt window.
2. To remove administrative access, run the following command:

   ```
   net localgroup "Administrators" "ssm-user" /delete
   ```
   -or-
   To restore administrative access, run the following command:

   ```
   net localgroup "Administrators" "ssm-user" /add
   ```

Use the Windows console to modify Administrator permissions

1. Connect to the instance and open the PowerShell or Command Prompt window.
2. From the command line, run `lusrmgr.msc` to open the **Local Users and Groups** console.
3. Open the **Users** directory, and then open **ssm-user**.
4. On the **Member Of** tab, do one of the following:
   - To remove administrative access, select **Administrators**, and then choose **Remove**.
   -or-
   - To restore administrative access, type **Administrators** in the text box, and then choose **Add**.
5. Choose **OK**.

**Step 8: (Optional) Enable SSH connections through Session Manager**

You can enable users in your AWS account to use the AWS CLI to establish Secure Shell (SSH) connections to instances using Session Manager. Users who connect using SSH can also copy files between their local machines and managed instances using Secure Copy Protocol (SCP). You can use this functionality to connect to instances without opening inbound ports or maintaining bastion hosts. You can also choose to explicitly disable SSH connections to your instances through Session Manager.

**Note**

Logging and auditing are not available for Session Manager sessions that connect through port forwarding or SSH. This is because SSH encrypts all session data, and Session Manager only serves as a tunnel for SSH connections.

To enable SSH connections through Session Manager

1. On the managed instance to which you want to enable SSH connections, do the following:
   - Ensure that SSH is running on the instance. (You can close inbound ports on the instance.)
   - Ensure that SSM Agent version 2.3.672.0 or later is installed on the instance.

   For information about installing or updating SSM Agent on an instance, see the following topics:
   - **Installing and configuring SSM Agent on Windows Server instances** (p. 65).
   - **Installing and configuring SSM Agent on EC2 instances for Linux** (p. 69)
   - **Install SSM Agent for a hybrid environment (Windows)** (p. 55)
• Install SSM Agent for a hybrid environment (Linux) (p. 52)

Note
To use Session Manager with on-premises servers and virtual machines (VMs) that you activated as managed instances, you must use the Advanced-Instances Tier. For more information about advanced instances, see Enabling the advanced-instances tier (p. 760).

2. On the local machine from which you want to connect to a managed instance using SSH, do the following:

• Ensure that version 1.1.23.0 or later of the Session Manager plugin is installed.

  For information about installing the Session Manager plugin, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

• Update the SSH configuration file to enable running a proxy command that starts a Session Manager session and transfer all data through the connection.

  **Linux**

  **Tip**
  The SSH configuration file is typically located at `~/.ssh/config`.

  Add the following to the configuration file on the local machine:

  ```
  # SSH over Session Manager
  host i-* mi-*
  ProxyCommand sh -c "aws ssm start-session --target %h --document-name AWS-StartSSHSession --parameters 'portNumber=%p'"
  ```

  **Windows**

  **Tip**
  The SSH configuration file is typically located at `C:\Users\username\.ssh\config`.

  Add the following to the configuration file on the local machine:

  ```
  # SSH over Session Manager
  host i-* mi-*
  ProxyCommand C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe "aws ssm start-session --target %h --document-name AWS-StartSSHSession --parameters portNumber=%p"
  ```

• Create or verify that you have a Privacy Enhanced Mail Certificate (a PEM file), or at minimum a public key, to use when establishing connections to managed instances. This must be a key that is already associated with the instance. For example, for an EC2 instance, the key-pair file you created or selected when you created the instance. (You specify the path to the certificate or key as part of the command to start a session. For information about starting a session using SSH, see Starting a session (SSH) (p. 820).)

**To enable SSH connections through Session Manager**

• Option 1: Open the IAM console at [https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/). In the navigation pane, choose **Policies**, and then update the permissions policy for the user or role you want to allow to start SSH connections through Session Manager. For example, prepare to modify the user quickstart policy you created in Quickstart end user policies for Session Manager (p. 784). Add the following element to the policy.
Option 2: Attach an inline policy to a user policy by using the AWS Management Console, the AWS CLI, or the AWS API.

Using the method of your choice, attach the policy statement in Option 1 to the policy for an AWS user, group, or role.

For information, see Adding and Removing IAM Identity Permissions in the IAM User Guide.

To disable SSH connections through Session Manager

- Option 1: Open the IAM console at https://console.aws.amazon.com/iam/. In the navigation pane, choose Policies, and then update the permissions policy for the user or role to block from starting Session Manager sessions. For example, prepare to modify the user quickstart policy you created in Quickstart end user policies for Session Manager (p. 784). Add the following element to the policy, or replace any permissions that allow a user to start a session:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor1",
      "Effect": "Deny",
      "Action": "ssm:StartSession",
      "Resource": "arn:aws:ssm:*:*:document/AWS-StartSSHSession"
    }
  ]
}
```

Option 2: Attach an inline policy to a user policy by using the AWS Management Console, the AWS CLI, or the AWS API.

Using the method of your choice, attach the policy statement in Option 1 to the policy for an AWS user, group, or role.

For information, see Adding and Removing IAM Identity Permissions in the IAM User Guide.

Working with Session Manager

You can use the AWS Systems Manager console, the Amazon EC2 console, or the AWS CLI to start sessions that connect you to the EC2 instances your system administrator has granted you access to using AWS Identity and Access Management (IAM) policies. Depending on your permissions, you can also view information about sessions, resume inactive sessions that have not timed out, and end sessions.

For more information about sessions, see What is a session? (p. 772)
Topics

- (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813)
- Start a session (p. 818)
- Terminate a session (p. 822)
- View session history (p. 822)

(Optional) Install the Session Manager Plugin for the AWS CLI

If you want to use the AWS CLI to start and end sessions that connect you to your managed instances, you must first install the Session Manager plugin on your local machine. The plugin can be installed on supported versions of Microsoft Windows, macOS, Linux, and Ubuntu Server.

Use the Latest Version of the Session Manager Plugin

The plugin is updated occasionally with enhanced functionality. We recommend that you regularly ensure you are using the latest version of the plugin. For more information, see Session Manager Plugin latest version and release history (p. 817).

Installation Prerequisite

AWS CLI version 1.16.12 or later must be installed on your local machine in order to use the Session Manager plugin.

Topics

- Install the Session Manager Plugin on Windows (p. 813)
- Install and uninstall the Session Manager Plugin on macOS (p. 814)
- Install Session Manager Plugin on Linux (p. 815)
- Install the Session Manager Plugin on Ubuntu Server (p. 815)
- Verify the Session Manager Plugin installation (p. 816)
- (Optional) enable Session Manager Plugin logging (p. 816)
- Session Manager Plugin latest version and release history (p. 817)

Install the Session Manager Plugin on Windows

You can install the Session Manager plugin on Microsoft Windows Vista or later using the standalone installer.

When updates are released, you must repeat the installation process to get the latest version of the Session Manager plugin.

Note

For best results, we recommend starting sessions on Windows clients using the Windows PowerShell application version 5 or later. On Microsoft Windows 10, the Command Prompt also provides reliable support for Session Manager operations.

To install the Session Manager plugin using the EXE installer

1. Download the installer using the following URL:

   https://s3.amazonaws.com/session-manager-downloads/plugin/latest/windows/
   SessionManagerPluginSetup.exe
2. Run the downloaded installer and follow the on-screen the instructions.

Leave the install location box blank to install the plugin to the default directory:

- C:\%PROGRAMFILES%\Amazon\SessionManagerPlugin\bin

3. Verify that the installation was successful. For information, see Verify the Session Manager Plugin installation (p. 816).

   **Note**
   If Windows is unable to find the executable, you might need to re-open the command prompt or add the installation directory to your PATH environment variable manually. For information, see the troubleshooting topic Session Manager Plugin not automatically added to command line path (Windows) (p. 828).

---

### Install and uninstall the Session Manager Plugin on macOS

You can install the Session Manager plugin on macOS using the bundled installer.

**Important**
The bundled installer does not support installing to paths that contain spaces.

**To install the Session Manager plugin using the bundled installer (macOS)**

1. Download the bundled installer:

   ```
   ```

2. Unzip the package:

   ```
   unzip sessionmanager-bundle.zip
   ```

3. Run the install command:

   ```
sudo ./sessionmanager-bundle/install -i /usr/local/sessionmanagerplugin -b /usr/local/bin/session-manager-plugin
   ```

   **Note**
The plugin requires Python 2.6.5 or later or Python 3.3. By default, the install script runs under the system default version of Python. If you have installed an alternative version of Python and want to use that to install the Session Manager plugin, run the install script with that version by absolute path to the Python executable. The following is an example.

   ```
sudo /usr/local/bin/python3.6 sessionmanager-bundle/install -i /usr/local/sessionmanagerplugin -b /usr/local/bin/session-manager-plugin
   ```

   The installer installs the Session Manager plugin at /usr/local/sessionmanagerplugin and creates the symlink `session-manager-plugin` in the /usr/local/bin directory. This eliminates the need to specify the install directory in the user's $PATH variable.

   To see an explanation of the `-i` and `-b` options, use the `-h` option:

   ```
   ./sessionmanager-bundle/install -h
   ```

4. Verify that the installation was successful. For information, see Verify the Session Manager Plugin installation (p. 816).
Install Session Manager Plugin on Linux

1. Download the Session Manager plugin RPM package:
   - 64-bit:
     ```bash
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_64bit/session-manager-plugin.rpm" -o "session-manager-plugin.rpm"
   ```
   - 32-bit:
     ```bash
   ```
   - ARM 64-bit:
     ```bash
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/linux_arm64/session-manager-plugin.rpm" -o "session-manager-plugin.rpm"
   ```

2. Run the install command:
   ```bash
   sudo yum install -y session-manager-plugin.rpm
   ```

3. Verify that the installation was successful. For information, see Verify the Session Manager Plugin installation (p. 816).

Note
If you ever want to uninstall the plugin, run the following two commands, one at a time:

```bash
sudo rm -rf /usr/local/sessionmanagerplugin

sudo rm /usr/local/bin/session-manager-plugin
```

Install the Session Manager Plugin on Ubuntu Server

1. Download the Session Manager plugin deb package:
   - 64-bit:
     ```bash
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-manager-plugin.deb" -o "session-manager-plugin.deb"
   ```
   - 32-bit:
     ```bash
   ```
   - ARM 64-bit:
     ```bash
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_arm64/session-manager-plugin.deb" -o "session-manager-plugin.deb"
   ```
2. Run the install command:

```
sudo dpkg -i session-manager-plugin.deb
```

3. Verify that the installation was successful. For information, see Verify the Session Manager Plugin installation (p. 816).

**Note**

If you ever want to uninstall the plugin, run `sudo dpkg -r session-manager-plugin`

### Verify the Session Manager Plugin installation

Run the following commands to verify that the Session Manager plugin installed successfully:

```
session-manager-plugin
```

If the installation was successful, the following message is returned:

```
The Session Manager plugin is installed successfully. Use the AWS CLI to start a session.
```

You can also test the installation by running the following command in the AWS CLI:

```
aws ssm start-session --target id-of-an-instance-you-have-permissions-to-access
```

(____Optional____) enable Session Manager Plugin logging

The Session Manager plugin includes an option to enable logging for sessions that you run. By default, logging is disabled.

If you enable logging, the Session Manager plugin creates log files for both application activity (`session-manager-plugin.log`) and errors (`errors.log`) on your local machine.

**Topics**

- Enable logging for the Session Manager Plugin (Windows) (p. 816)
- Enable logging for the Session Manager Plugin (Linux and macOS) (p. 817)

### Enable logging for the Session Manager Plugin (Windows)

1. Locate the `seelog.xml.template` file for the plugin.

   The default location is `C:\Program Files\Amazon\SessionManagerPlugin\seelog.xml.template`.

2. Change the name of the file to `seelog.xml`.

3. Open the file and change `minlevel="off"` to `minlevel="info"` or `minlevel="debug"`.

   **Note**
   
   By default, log entries about opening a data channel and reconnecting sessions are recorded at the INFO level. Data flow (packets and acknowledgement) entries are recorded at the DEBUG level.

4. Change other configuration options you want to modify. Options you can change include:
• **Debug level**: You can change the debug level from `fmtinfo` to `fmtdebug`.

• **Log file options**: You can make changes to the log file options, including where the logs are stored, with the exception of the log file names.

  **Important**
  Do not change the file names or logging will not work correctly.

```xml
<rollingfile type="size" filename="C:\%PROGRAMDATA%\Amazon\SessionManagerPlugin\Logs\session-manager-plugin.log" maxsize="30000000" maxrolls="5"/>
<filter levels="error,critical" formatid="fmterror">
  <rollingfile type="size" filename="C:\%PROGRAMDATA%\Amazon\SessionManagerPlugin\Logs\errors.log" maxsize="10000000" maxrolls="5"/>
</filter>
```

5. Save the file.

**Enable logging for the Session Manager Plugin (Linux and macOS)**

1. Locate the `seelog.xml.template` file for the plugin.

   The default location is `/usr/local/sessionmanagerplugin/seelog.xml.template`.

2. Change the name of the file to `seelog.xml`.

3. Open the file and change `minlevel="off"` to `minlevel="info"` or `minlevel="debug"`.

   **Note**
   By default, log entries about opening data channels and reconnecting sessions are recorded at the INFO level. Data flow (packets and acknowledgement) entries are recorded at the DEBUG level.

4. Change other configuration options you want to modify. Options you can change include:

   • **Debug level**: You can change the debug level from `fmtinfo` to `fmtdebug`

   • **Log file options**: You can make changes to the log file options, including where the logs are stored, with the exception of the log file names.

     **Important**
     Do not change the file names or logging will not work correctly.

```xml
<rollingfile type="size" filename="/usr/local/sessionmanagerplugin/logs/session-manager-plugin.log" maxsize="30000000" maxrolls="5"/>
<filter levels="error,critical" formatid="fmterror">
  <rollingfile type="size" filename="/usr/local/sessionmanagerplugin/logs/errors.log" maxsize="10000000" maxrolls="5"/>
</filter>
```

**Important**
If you use the specified default directory for storing logs, you must either run session commands using `sudo` or give the directory where the plugin is installed full read and write permissions. To bypass these restrictions, change the location where logs are stored.

5. Save the file.

**Session Manager Plugin latest version and release history**

Your local machine must be running a supported version of the Session Manager plugin. If you are running an earlier version, your Session Manager operations might not succeed.

The current minimum supported version is 1.1.17.0.
To see if you have the latest version, run the following command in the AWS CLI:

**Note**
The command returns results only if the plugin is located in the default installation directory for your operating system type. You can also check the version in the contents of the `VERSION` file in the directory where you have installed the plugin.

```
session-manager-plugin --version
```

The following table lists all releases of the Session Manager plugin and the features and enhancements included with each version.

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.61.0</td>
<td>April 17, 2020</td>
<td><strong>Enhancement</strong>: Added ARM support for Linux and Ubuntu Server.</td>
</tr>
<tr>
<td>1.1.54.0</td>
<td>January 6, 2020</td>
<td><strong>Bug fix</strong>: Handle race condition scenario of packets being dropped when the Session Manager plugin is not ready.</td>
</tr>
<tr>
<td>1.1.50.0</td>
<td>November 19, 2019</td>
<td><strong>Enhancement</strong>: Added support for forwarding a port to a local unix socket.</td>
</tr>
<tr>
<td>1.1.35.0</td>
<td>November 7, 2019</td>
<td><strong>Enhancement</strong>: (Port forwarding sessions only) Send a TerminateSession command to SSM Agent when the local user presses Ctrl+C.</td>
</tr>
<tr>
<td>1.1.33.0</td>
<td>September 26, 2019</td>
<td><strong>Enhancement</strong>: (Port forwarding sessions only) Send a disconnect signal to the server when the client drops the TCP connection.</td>
</tr>
<tr>
<td>1.1.31.0</td>
<td>September 6, 2019</td>
<td><strong>Enhancement</strong>: Update to keep port forwarding session open until remote server closes the connection.</td>
</tr>
<tr>
<td>1.1.26.0</td>
<td>July 30, 2019</td>
<td><strong>Enhancement</strong>: Update to limit the rate of data transfer during a session.</td>
</tr>
<tr>
<td>1.1.23.0</td>
<td>July 9, 2019</td>
<td><strong>Enhancement</strong>: Added support for running SSH sessions using Session Manager.</td>
</tr>
<tr>
<td>1.1.17.0</td>
<td>April 4, 2019</td>
<td><strong>Enhancement</strong>: Added support for further encryption of session data using AWS Key Management Service (AWS KMS).</td>
</tr>
<tr>
<td>1.0.37.0</td>
<td>September 20, 2018</td>
<td><strong>Enhancement</strong>: Bug fix for Windows version.</td>
</tr>
<tr>
<td>1.0.0.0</td>
<td>September 11, 2018</td>
<td>Initial release of the Session Manager plugin.</td>
</tr>
</tbody>
</table>

### Start a session

You can use the AWS Systems Manager console, the Amazon EC2 console, the AWS CLI, or SSH to start a session.

**Topics**
- Starting a session (Systems Manager console) (p. 819)
- Starting a session (Amazon EC2 console) (p. 819)
- Starting a session (AWS CLI) (p. 820)
- Starting a session (SSH) (p. 820)
- Starting a session (port forwarding) (p. 821)
• Starting a session (interactive commands) (p. 821)

Starting a session (Systems Manager console)

You can use the AWS Systems Manager console to start a session with an instance in your account.

Note
Before attempting to start a session, ensure that the necessary setup steps for Session Manager have been completed. For information, see Getting started with Session Manager (p. 773).

To start a session (Systems Manager console)

2. In the navigation pane, choose Session Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Session Manager in the navigation pane.
3. Choose Start session.
4. For Target instances, choose the option button to the left of the instance you want to connect to.
   If an instance you want to connect to is not in the list, or is listed but an error message reports, "The instance you selected is not configured to use Session Manager," see Instance not available or not configured for Session Manager (p. 827) for troubleshooting steps.
5. Choose Start session.

After the connection is made, you can run bash commands (Linux) or PowerShell commands (Windows) as you would through any other connection type.

Starting a session (Amazon EC2 console)

You can use the Amazon Elastic Compute Cloud (Amazon EC2) console to start a session with an instance in your account.

Note
If you receive an error that you're not authorized to perform one or more Systems Manager actions (ssm:command-name, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to start sessions from the Amazon EC2 console. If you are an administrator, see Quickstart default IAM policies for Session Manager (p. 784) for more information.

To start a session (Amazon EC2 console)

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Instances
3. Select the instance and choose Connect.
4. For Connection method, choose Session Manager.
5. Choose Connect.

After the connection is made, you can run bash commands (Linux) or PowerShell commands (Windows) as you would through any other connection type.
Starting a session (AWS CLI)

To start a session using the AWS CLI, run the following command:

```
aws ssm start-session --target instance-id
```

`instance-id` represents the ID of an instance configured for use with AWS Systems Manager and its Session Manager capability, such as `i-02573cafcfEXAMPLE`.

For information about other options you can use with the `start-session` command, see `start-session` in the AWS Systems Manager section of the AWS CLI Command Reference.

Starting a session (SSH)

SSH Connection Requirements

Take note of the following requirements and limitations for session connections using SSH.

- Your target instance must be configured to support SSH connections. For more information, see (Optional) Enable SSH connections through Session Manager (p. 810).
- You must use the user on the instance associated with the Privacy Enhanced Mail (PEM) certificate, not the `ssm-user` account that is used for other types of session connections. For example, on EC2 instances for Linux, the default user is `ec2-user`. For information about identifying the default user for each instance type, see Get Information About Your Instance in the Amazon EC2 User Guide for Linux Instances.
- Logging and auditing are not available for Session Manager sessions that connect through port forwarding or SSH. This is because SSH encrypts all session data, and Session Manager only serves as a tunnel for SSH connections.

```
ssh -i /path/my-key-pair.pem username@instance-id
```

`/path/my-key-pair.pem` represents the path to the PEM certificate that is associated with the instance. For example, for an EC2 instance, the key-pair file you created or selected when you created the instance.

`username@instance-id` represents the default user name for your instance type, and the instance ID, such as `ec2-user@i-02573cafcfEXAMPLE`.

**Tip**

When you start a session using SSH, you can copy local files to the target instance using the following command format.

```
scp -i /path/my-key-pair.pem /path/SampleFile.txt username@instance-id:~
```
For information about other options you can use with the `start-session` command, see `start-session` in the AWS Systems Manager section of the AWS CLI Command Reference.

### Starting a session (port forwarding)

To start a port forwarding session, run the following command from the CLI:

**Note**
Before attempting to start a session, ensure that the necessary setup steps for Session Manager have been completed. For information, see Getting started with Session Manager (p. 773).

To use the AWS CLI to run session commands, the Session Manager plugin must also be installed on your local machine. For information, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

**Linux**

```bash
aws ssm start-session \
--target instance-id \
--document-name AWS-StartPortForwardingSession \
--parameters '{"portNumber": ["80"], "localPortNumber": ["56789"]}'
```

**Windows**

```bash
aws ssm start-session ^
--target instance-id ^
--document-name AWS-StartPortForwardingSession ^
--parameters portNumber="3389",localPortNumber="56789"
```

*instance-id* represents the ID of an instance configured for use with AWS Systems Manager and its Session Manager capability, such as *i-02573cafcfEXAMPLE*.

*portNumber* represents the remote port on the instance where traffic should be redirected to, such as 3389. If this parameter is not specified, Session Manager assumes 80 as the default remote port.

*localPortNumber* represents the local port on the client where traffic should be redirected to, such as 56789.

For information about other options you can use with the `start-session` command, see `start-session` in the AWS Systems Manager section of the AWS CLI Command Reference.

### Starting a session (interactive commands)

To start an Interactive Command session, run the following command:

**Note**
Before attempting to start a session, ensure that the necessary setup steps for Session Manager have been completed. For information, see Getting started with Session Manager (p. 773).

To use the AWS CLI to run session commands, the Session Manager plugin must also be installed on your local machine. For information, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

```bash
aws ssm start-session --target instance-id --document-name CustomCommandSessionDocument --parameters '{"logpath": ["/var/log/amazon/ssm/amazon-ssm-agent.log"]}'
```

*instance-id* represents the ID of an instance configured for use with AWS Systems Manager and its Session Manager capability, such as *i-02573cafcfEXAMPLE*. 

821
For information about other options you can use with the `start-session` command, see `start-session` in the AWS Systems Manager section of the AWS CLI Command Reference.

Related Content

Port Forwarding Using AWS Systems Manager Session Manager on the AWS News Blog.

Terminate a session

You can use the AWS Systems Manager console or the AWS CLI to end a session that you started to connect to an instance in your account. If there is no user activity after 20 minutes, a session is ended. After a session is ended, it can't be resumed.

Topics

- Terminating a session (console) (p. 822)
- Terminating a session (AWS CLI) (p. 822)

Terminating a session (console)

You can use the AWS Systems Manager console to end a session with an instance in your account.

To end a session (console)

2. In the navigation pane, choose Session Manager.
3. For Sessions, choose the option button to the left of the session you want to end.
4. Choose Terminate.

Terminating a session (AWS CLI)

To end a session using the AWS CLI, run the following command:

```
aws ssm terminate-session --session-id session-id
```

`session-id` represents the ID of an active Session Manager session that you want to end permanently.

For more information about the `terminate-session` command, see `terminate-session` in the AWS Systems Manager section of the AWS CLI Command Reference.

View session history

You can use the AWS Systems Manager console or the AWS CLI to view information about sessions in your account. In the console, you can view session details such the following:

- The ID of the session
- Which user connected to an instance through a session
• The ID of the instance
• When the session began and ended
• The status of the session
• The location specified for storing session logs (if enabled)

Using the AWS CLI, you can view a list of sessions in your account, but not the additional details that are available in the console.

For information about auditing and logging session history information, see Auditing and logging session activity (p. 823).

Topics
• Viewing session history (console) (p. 823)
• Viewing session history (AWS CLI) (p. 823)

Viewing session history (console)

You can use the AWS Systems Manager console to view details about the sessions in your account.

To view session history (console)
2. In the navigation pane, choose Session Manager.
3. Choose Start session.
4. Choose the Session history tab.

Viewing session history (AWS CLI)

To view a list of sessions in your account using the AWS CLI, run the following command:

```
aws ssm describe-sessions --state History
```

Note
This command returns only results for connections to targets initiated using Session Manager. It does not list connections made through other means, such as Remote Desktop Protocol (RDP) or the Secure Shell Protocol (SSH).

For information about other options you can use with the describe-sessions command, see describe-sessions in the AWS Systems Manager section of the AWS CLI Command Reference.

Auditing and logging session activity

In addition to providing information about current and completed sessions in the Systems Manager console, Session Manager provides you with options for auditing and logging session activity in your AWS account. This allows you to do the following:

• Create and store session logs for archival purposes.
• Generate a report showing details of every connection made to your instances using Session Manager over the past 30 days.
• Generate notifications of session activity in your AWS account, such as Amazon Simple Notification Service (Amazon SNS) notifications.
Automatically initiate another action on an AWS resource as the result of session activity, such as running an AWS Lambda function, starting an AWS CodePipeline pipeline, or running an AWS Systems Manager Run Command document.

**Important**
Take note of the following requirements and limitations for Session Manager.

- If you are using Windows Server 2012 or earlier, the data in your logs might not be formatted optimally. We recommend using Windows Server 2012 R2 and later for optimal log formats.
- If you are using Linux instances, ensure that the screen utility is installed. If it is not, your log data might be truncated. On Amazon Linux, Amazon Linux 2, and Ubuntu Server, the screen utility is installed by default. To install screen manually, depending on your version of Linux, run either `sudo yum install screen` or `sudo apt-get install screen`.
- Logging and auditing are not available for Session Manager sessions that connect through port forwarding or SSH. This is because SSH encrypts all session data, and Session Manager only serves as a tunnel for SSH connections.

For more information about the permissions required to use Amazon S3 or Amazon CloudWatch Logs for logging session data, see [Creating an instance profile with permissions for Session Manager and Amazon S3 and CloudWatch Logs (console)](p. 780).

Refer to the following topics for more information about auditing and logging options for Session Manager.

### Audit session activity using AWS CloudTrail

AWS CloudTrail captures session API calls through the Systems Manager console, the AWS CLI, and the Systems Manager SDK. The information can be viewed on the CloudTrail console or stored in a specified Amazon S3 bucket. One bucket is used for all CloudTrail logs for your account.

For more information, see [Logging AWS Systems Manager API calls with AWS CloudTrail](p. 1144).

### Logging session data using Amazon S3 (console)

You can choose to store session log data in a specified S3 bucket for auditing purposes. The default option is for logs to be sent to an encrypted S3 bucket. Encryption is performed using the key specified for the bucket, either an AWS Key Management Service (AWS KMS) key or an Amazon S3 Server-Side Encryption (SSE) key (AES-256).

**Important**
When you use virtual hosted–style buckets with Secure Sockets Layer (SSL), the SSL wildcard certificate only matches buckets that don't contain periods. To work around this, use HTTP or write your own certificate verification logic. We recommend that you do not use periods ("." in bucket names when using virtual hosted–style buckets.

#### S3 Bucket Encryption

In order to send logs to your S3 bucket with encryption, encryption must be enabled on the bucket. For more information about S3 bucket encryption, see [Amazon S3 Default Encryption for S3 Buckets](p. 780).

#### Customer-managed CMK

If you are using an AWS KMS customer master key (CMK) that you manage yourself (a customer-managed CMK) to encrypt your bucket, then the IAM instance profile attached to your instances must have explicit permissions to read the CMK. If you use an AWS-managed CMK, the instance does not require this explicit permission. For more information about providing the instance profile with access to use the CMK, see [Allows Key Users to Use the CMK](p. 1144) in the [AWS Key Management Service Developer Guide](p. 780).
Follow these steps to configure Session Manager to store session logs in an Amazon S3 bucket.

**Note**
You can also use the AWS CLI to specify or change the S3 bucket that session data is sent to. For information, see Update Session Manager preferences (command line) (p. 802).

### To log session data using Amazon S3 (console)

2. In the navigation pane, choose **Session Manager**.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Select the check box next to **S3 bucket**.
5. (Optional) If you do not want to encrypt the log data that is sent to the S3 bucket, clear the check box next to **Encrypt log data**. Otherwise, log data is encrypted using the server-side encryption key specified for the bucket. You must also clear the check box if encryption is not enabled on the bucket.
6. For **S3 bucket name**, select one of the following:
   - **Choose a bucket name from the list**: Select an S3 bucket that has already been created in your account to store session log data.
   - **Enter a bucket name in the text box**: Enter the name of an S3 bucket that has already been created in your account to store session log data.
7. (Optional) For **S3 key prefix**, enter the name of an existing or new folder to store logs in the selected bucket.
8. Choose **Save**.

For more information about working with Amazon S3 and S3 buckets, see the [Amazon Simple Storage Service Getting Started Guide](https://docs.aws.amazon.com/AmazonS3/latest.GetUserGuide) and the [Amazon Simple Storage Service Console User Guide](https://docs.aws.amazon.com/AmazonS3/latest/ConsoleGuide).

### Logging session data using Amazon CloudWatch Logs (console)

Amazon CloudWatch Logs lets you monitor, store, and access log files from various AWS services. You can send session log data to a CloudWatch Logs log group for auditing purposes. The default option is for log data to be sent with encryption using your AWS KMS key, but you can send the data to your log group with or without encryption.

Follow these steps to configure Session Manager to send session log data to a CloudWatch Logs log group.

**Note**
You can also use the AWS CLI to specify or change the CloudWatch Logs log group that session data is sent to. For information, see Update Session Manager preferences (command line) (p. 802).

### To log session data using Amazon CloudWatch Logs (console)

2. In the navigation pane, choose **Session Manager**.
3. Choose the **Preferences** tab, and then choose **Edit**.
4. Select the check box next to **CloudWatch logs**.
5. (Optional) If you do not want to encrypt the log data that is sent to CloudWatch Logs, clear the check box next to **Encrypt log data**. Otherwise, log data is encrypted using the server-side encryption key specified for the log group. You must also clear the check box if encryption is not enabled on the log group.

6. For **CloudWatch logs**, to specify the existing CloudWatch Logs log group in your AWS account to upload session logs to, select one of the following:
   - **Choose a log group from the list**: Select a log group that has already been created in your account to store session log data.
   - **Enter a log group name in the text box**: Enter the name of a log group that has already been created in your account to store session log data.

7. Choose **Save**.

For more information about working with CloudWatch Logs, see the [*Amazon CloudWatch Logs User Guide*](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/index.html).

### Monitoring session activity using Amazon CloudWatch Events (console)

CloudWatch Events lets you set up rules to detect when changes happen to AWS resources. You can create a rule to detect when a user in your organization starts or ends a session, and then, for example, receive a notification through Amazon SNS about the event.

CloudWatch Events support for Session Manager relies on records of API actions that were recorded by CloudTrail. (You can use CloudTrail integration with CloudWatch Events to respond to most AWS Systems Manager events.)

The following steps outline how to trigger notifications through Amazon Simple Notification Service (Amazon SNS) when a Session Manager API event occurs, such as `StartSession`.

#### To monitor session activity using Amazon CloudWatch Events (console)

1. Create an Amazon SNS topic to use for sending notifications when the Session Manager event occurs that you want to track.

   For more information, see [Create a Topic](https://docs.aws.amazon.com/AmazonSNS/latest/dg/SNS-Tutorial.html) in the [*Amazon Simple Notification Service Developer Guide*](https://docs.aws.amazon.com/AmazonSNS/latest/dg/SNS-Tutorial.html).

2. Create a CloudWatch Events rule to invoke the Amazon SNS target for the type of Session Manager event you want to track.

   For information about how to create the rule, see [Creating a CloudWatch Events Rule That Triggers on an Event](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Creating-a-Rule.html) in the [*Amazon CloudWatch Events User Guide*](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/Creating-a-Rule.html).

As you follow the steps to create the rule, make the following selections:

- For **Service Name**, choose **EC2 Simple Systems Manager (SSM)**.
- For **Event Type**, choose **AWS API Call via CloudTrail**.
- Choose **Specific operation(s)**, and then enter the Session Manager command or commands (one at a time) you want to receive notifications for. You can choose `StartSession`, `ResumeSession`, and `TerminateSession`. (CloudWatch Events doesn't support `Get*`, `List*`, and `Describe*` commands.)
- For **Targets**, choose **SNS topic**. For **Topic**, choose the name of the Amazon SNS topic you created in Step 1.

For more information, see the [*Amazon CloudWatch Events User Guide*](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/index.html) and the [*Amazon Simple Notification Service Getting Started Guide*](https://docs.aws.amazon.com/AmazonSNS/latest/dg/SNS-GetStarted.html).
Troubleshooting Session Manager

Use the following information to help you troubleshoot problems with Session Manager.

Topics
- No permission to start a session (p. 827)
- No permission to change session preferences (p. 827)
- Instance not available or not configured for Session Manager (p. 827)
- Session Manager Plugin not found (p. 828)
- Session Manager Plugin not automatically added to command line path (Windows) (p. 828)
- TargetNotConnected (p. 829)
- Blank screen displays after starting a session (p. 829)

No permission to start a session

Problem: You try to start a session, but the system tells you that you do not have the necessary permissions.

Solution: A system administrator has not granted you IAM policy permissions for starting Session Manager sessions. For information, see Control user session access to instances (p. 782).

No permission to change session preferences

Problem: You try to update global session preferences for your organization, but the system tells you that you do not have the necessary permissions.

Solution: A system administrator has not granted you IAM policy permissions for setting Session Manager preferences. For information, see Grant or deny a user permissions to update Session Manager preferences (p. 796).

Instance not available or not configured for Session Manager

Problem 1: You want to start a session on the Start a session console page, but an instance is not in the list.

Solution: The instance you want to connect to might not have been configured to use with the AWS Systems Manager service. To use an instance with Systems Manager, an IAM instance profile that gives Systems Manager permission to perform actions on your instances must be attached to the instance. For information, see Create an IAM instance profile for Systems Manager (p. 30).

Note
If SSM Agent is already running on an instance when you attach the IAM instance profile, you might need to restart the agent before the instance is listed on the Start a session console page.

Problem 2: An instance you want to connect is in the list on the Start a session console page, but the page reports that “The instance you selected is not configured to use Session Manager.”

Solution A: The instance has been configured for use with the AWS Systems Manager service, but the IAM instance profile attached to the instance might not include permissions for the Session Manager capability. For information, see Verify or Create an IAM Instance Profile with Session Manager Permissions (p. 776).
• **Solution B:** The instance is not running a version of SSM Agent that supports Session Manager. Update SSM Agent on the instance to version 2.3.68.0 or later.

Update SSM Agent manually on an instance by following the steps in Install and configure SSM Agent on EC2 instances for Windows Server (p. 65) or Manually install SSM Agent on EC2 instances for Linux (p. 69), depending on the operating system.

Alternatively, use the Run Command document `AWS-UpdateSSMAgent` to update the agent version on one or more instances at a time. For information, see Update SSM Agent by using Run Command (p. 835).

**Tip**
To always keep your agent up-to-date, we recommend updating SSM Agent to the latest version on an automated schedule that you define using either of the following methods:

- Run `AWS-UpdateSSMAgent` as part of a State Manager association. For information, see Automatically update SSM Agent (CLI) (p. 908).
- Run `AWS-UpdateSSMAgent` as part of a maintenance window. For information about working with maintenance windows, see Working with maintenance windows (console) (p. 634) and Tutorial: Create and configure a maintenance window (AWS CLI) (p. 642).

• **Solution C:** The instance is in a Virtual Private Cloud (VPC), but an `ssmmessages` endpoint has not been created in the VPC. An `ssmmessages` endpoint in the format `com.amazonaws.region.ssmmessages` is required if you are connecting to your instances through a secure data channel using Session Manager. For more information, see Creating VPC endpoints for Systems Manager (p. 39).

• **Solution D:** The instance has limited available CPU or memory resources. Though your instance may otherwise be functional, if the instance does not have enough available resources, you can't establish a session. For more information, see Troubleshooting an Unreachable Instance.

### Session Manager Plugin not found

To use the AWS CLI to run session commands, the Session Manager plugin must also be installed on your local machine. For information, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

### Session Manager Plugin not automatically added to command line path (Windows)

When you install the Session Manager plugin on Windows, the `session-manager-plugin` executable should be automatically added to your operating system's `PATH` environment variable. If the command failed after you ran it to check whether the Session Manager plugin installed correctly (`aws ssm start-session --target instance-id`), you might need to set it manually using the following procedure.

**To modify your PATH variable (Windows)**

1. Press the Windows key and enter **environment variables**.
2. Choose **Edit environment variables for your account**.
3. Choose **PATH** and then choose **Edit**.
4. Add paths to the **Variable value** field, separated by semicolons, as shown in this example: `C:\existing\path;C:\new\path`

   `C:\existing\path` represents the value already in the field. `C:\new\path` represents the path you want to add, as shown in these examples.
• **64-bit machines**: C:\Program Files\Amazon\SessionManagerPlugin\bin\n
• **32-bit machines**: C:\Program Files (x86)\Amazon\SessionManagerPlugin\bin\n
5. Choose OK twice to apply the new settings.

6. Close any running command prompts and re-open.

**TargetNotConnected**

**Problem**: You try to start a session, but the system returns the error message, "An error occurred (TargetNotConnected) when calling the StartSession operation: **InstanceID** is not connected."

• **Solution**: This error is returned when the specified target instance for the session is not fully configured for use with Session Manager. For information, see Getting started with Session Manager (p. 773).

**Blank screen displays after starting a session**

**Problem**: You start a session and Session Manager displays a blank screen.

• **Solution A**: This issue can occur when the root volume on the instance is full. Due to lack of disk space, SSM Agent on the instance stops working. To resolve this issue, use Amazon CloudWatch to collect metrics and logs from the operating systems. For information, see Monitoring memory and disk metrics for Amazon EC2 Linux instances or Monitoring memory and disk metrics for Amazon EC2 Windows instances.

• **Solution B**: A blank screen might display if you've accessed the console using a link that includes a mismatched endpoint and Region pair. For example, in the following console URL, **us-west-2** is the specified endpoint, but **us-west-1** is the specified AWS Region:

```plaintext
https://us-west-2.console.aws.amazon.com/systems-manager/session-manager/sessions?
region=us-west-1
```

• **Solution C**: The instance is connecting to Systems Manager using VPC endpoints, and your Session Manager preferences write session output to an Amazon S3 bucket, but an s3 gateway endpoint does not exist in the VPC. An s3 endpoint in the format **com.amazonaws.region.s3** is required if your instances are connecting to Systems Manager using VPC endpoints, and your Session Manager preferences write session output to an Amazon S3 bucket. For more information, see Creating VPC endpoints for Systems Manager (p. 39).

**AWS Systems Manager Run Command**

AWS Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A **managed instance** is any EC2 instance or on-premises machine in your hybrid environment that has been configured for Systems Manager. Run Command enables you to automate common administrative tasks and perform ad hoc configuration changes at scale. You can use Run Command from the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.

Administrators use Run Command to perform the following types of tasks on their managed instances: install or bootstrap applications, build a deployment pipeline, capture log files when an instance is terminated from an Auto Scaling group, and join instances to a Windows domain, to name a few.

**Getting Started**
The following table includes information to help you get started with Run Command.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Manager prerequisites (p. 14)</td>
<td>(Required) Verify that your instances meet the minimum requirements for Run Command, configure required roles, and install SSM Agent.</td>
</tr>
<tr>
<td>Setting up AWS Systems Manager for hybrid environments (p. 43)</td>
<td>(Optional) Register on-premises servers and VMs with AWS so that you can manage them using Run Command.</td>
</tr>
<tr>
<td>Running commands using Systems Manager Run Command (p. 833)</td>
<td>Learn how to run a command from the EC2 console and how to run commands to a fleet of managed instances.</td>
</tr>
<tr>
<td>Run Command walkthroughs (p. 850)</td>
<td>Learn how to run commands using either AWS Tools for Windows PowerShell or the AWS CLI.</td>
</tr>
</tbody>
</table>

Related content

- Remotely Run Commands on an EC2 Instance (10 minute tutorial)
- Systems Manager service quotas in the Amazon Web Services General Reference
- AWS Systems Manager API Reference

Contents

- Setting up Run Command (p. 830)
- Running commands using Systems Manager Run Command (p. 833)
- Handling exit codes with scripts (p. 843)
- Understanding command statuses (p. 845)
- Run Command walkthroughs (p. 850)
- Troubleshooting Systems Manager Run Command (p. 862)

Setting up Run Command

Before you can manage instances by using Run Command, you must configure an AWS Identity and Access Management (IAM) user policy for any user who will run commands. For more information, see Create non-Admin IAM users and groups for Systems Manager (p. 25).

You must also create an IAM instance profile role for any instance that will process commands and attach it to those instances. For more information, see Create an IAM instance profile for Systems Manager (p. 30) and Attach an IAM instance profile to an EC2 instance (p. 35).

We also strongly recommend completing the following optional setup tasks to help minimize the security posture and day-to-day management of your instances.

Monitor command executions using Amazon CloudWatch Events

You can use Amazon CloudWatch Events to log command execution status changes. You can create a rule that runs whenever there is a state transition, or when there is a transition to one or more states that are of interest. You can also specify Run Command as a target action when a CloudWatch event occurs. For more information, see Configuring CloudWatch Events for Run Command (p. 1149).
Monitor command executions using Amazon CloudWatch Logs

You can configure Run Command to periodically send all command output and error logs to a CloudWatch Logs log group. You can monitor these output logs in near real-time, search for specific phrases, values, or patterns, and create alarms based on the search. For more information, see Configuring Amazon CloudWatch Logs for Run Command (p. 1146).

Restrict Run Command access to specific instances

You can restrict which of your managed instances commands can be run on by creating an IAM user policy that includes a condition that the user can only run commands on instances that are tagged with specific Amazon EC2 tags. For more information, see the following topic, Restricting Run Command access based on instance tags (p. 831).

Restricting Run Command access based on instance tags

You can restrict command execution to specific instances by creating an IAM user policy that includes a condition that the user can only run commands on instances that are tagged with specific Amazon EC2 tags. In the following example, the user is allowed to use Run Command (Effect: Allow, Action: ssm:SendCommand) by using any SSM document (Resource: arn:aws:ssm:*:*:document/*) on any instance (Resource: arn:aws:ec2:*:*:instance/*) with the condition that the instance is a Finance WebServer (ssm:resourceTag/Finance: WebServer). If the user sends a command to an instance that is not tagged or that has any tag other than Finance: WebServer, the execution results show AccessDenied.

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

You can create IAM policies that enable a user to run commands on instances that are tagged with multiple tags. The following policy enables the user to run commands on instances that have two tags. If a user sends a command to an instance that is not tagged with both of these tags, the execution results show AccessDenied.
You can also create IAM policies that enable a user to run commands on multiple groups of tagged instances. The following policy enables the user to run commands on either group of tagged instances, or both groups.

```json
{
  "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": "*",
      "Condition": {
        "StringLike": {
          "ssm:resourceTag/tag_key1": [
            "tag_value1"
          ]
        }
      }
    }
  ]
}
```
For more information about creating IAM user policies, see Managed Policies and Inline Policies in the IAM User Guide. For more information about tagging instances, see Tagging your Amazon EC2 resources in the Amazon EC2 User Guide for Linux Instances (content applies to Windows and Linux instances).

Running commands using Systems Manager Run Command

This section includes information about how to send commands from the AWS Systems Manager console, and how to send commands to a fleet of instances by using the Targets parameter with EC2 tags. This section also includes information about how to cancel a command.

For information about how to send commands using Windows PowerShell, see Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command (p. 853) or the examples in the AWS Systems Manager section of the AWS Tools for PowerShell Cmdlet Reference. For information about how to send commands using the AWS CLI, see the Walkthrough: Use the AWS CLI with Run Command (p. 850) or the examples in the SSM CLI Reference.

Important
If this is your first time using Run Command, we recommend executing commands against a test instance or an instance that is not being used in a production environment.

Contents
- Running commands from the console (p. 834)
• Running PowerShell scripts on Linux instances (p. 837)
• Running commands using the document version parameter (p. 838)
• Using targets and rate controls to send commands to a fleet (p. 840)
• Canceling a command (p. 843)

Running commands from the console

You can use Run Command from the console to configure instances without having to login to each instance. This topic includes an example that shows how to update SSM Agent (p. 835) on an instance by using Run Command.

Before You Begin

Before you send a command using Run Command, verify that your instances meet Systems Manager requirements (p. 14).

To send a command using Run Command

2. In the navigation pane, choose Run Command.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Run Command.
3. Choose Run command.
4. In the Command document list, choose a Systems Manager document.
5. In the Command parameters section, specify values for required parameters.
6. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.
   
   Note
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.
7. For Other parameters:
   • For Comment, type information about this command.
   • For Timeout (seconds), specify the number of seconds for the system to wait before failing the overall command execution.
8. (Optional) For Rate control:
   • For Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.
      
      Note
      If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
   • For Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.
9. (Optional) For Output options, to save the command output to a file, select the Write command output to an S3 bucket box. Type the bucket and prefix (folder) names in the boxes.
**Running commands**

**Note**
The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

   For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

11. Choose **Run**.

For information about canceling a command, see the section called "Canceling a command" (p. 843).

**Rerunning commands**

Systems Manager includes two options to help you rerun a command from the **Run Command** page in the AWS Systems Manager console.

- **Rerun**: This button enables you to run the same command without making changes to it.
- **Copy to new**: This button copies the settings of one command to a new command and gives you the option to edit those settings before you run it.

**To rerun a command**

2. In the navigation pane, choose **Run Command**.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon to open the navigation pane, and then choose **Run Command**.
3. Choose a command to rerun. You can rerun a command immediately after executing it from the command details page. Or, you can choose a command that you previously executed from the **Command history** tab.
4. Choose either **Rerun** to run the same command without changes, or choose **Copy to new** to edit the command settings before you run it.

**Update SSM Agent by using Run Command**

The following procedure describes how to quickly update SSM Agent running on your Windows Server and Linux instances. You can update to either the latest version or downgrade to an older version. When you run the command, the system downloads the version from AWS, installs it, and then uninstalls the version that existed before the command was run. If an error occurs during this process, the system rolls back to the version on the server before the command was run and the command status shows that the command failed.

**Note**
To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

**To update SSM Agent using Run Command**

2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Run Command**.

3. Choose **Run command**.

4. In the **Command document** list, choose **AWS-UpdateSSMAgent**.

5. In the **Command parameters** section, specify values for the following parameters, if you want:

   a. (Optional) For **Version**, type the version of SSM Agent to install. You can install older versions of the agent. If you do not specify a version, the service installs the latest version.

   b. (Optional) For **Allow Downgrade**, choose **true** to install an earlier version of SSM Agent. If you choose this option, you must specify the earlier version number. Choose **false** to install only the newest version of the service.

6. In the **Targets** section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

   **Note**
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see **Where are my instances? (p. 862)** for troubleshooting tips.

7. For **Other parameters**:

   a. For **Comment**, type information about this command.

   b. For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. (Optional) For **Rate control**:

   a. For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

      **Note**
      If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

   b. For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

9. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.

   **Note**
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see **Create an IAM instance profile for Systems Manager (p. 30)**. In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

    For more information about configuring Amazon SNS notifications for Run Command, see **Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151)**.

11. Choose **Run**.
Running PowerShell scripts on Linux instances

Using the `aws:runPowerShellScript` plugin or the `AWS-RunPowerShellScript` command document, along with PowerShell Core, you can run PowerShell scripts on Linux instances. This can be useful for systems administrators who are familiar with PowerShell and prefer it to other scripting languages.

**Before You Begin**

Connect to your Linux instance and follow the PowerShell Core installation procedure for the appropriate operating system.

**Note**

Many PowerShell commands (cmdlets) are not available on Linux. To see which commands are available, use the `Get-Command` cmdlet after starting PowerShell using the `pwsh` command on your Linux instance. For more information, see Get-Command.

The following procedure describes how to run a PowerShell script on a Linux instance using the console.

**To run a PowerShell script on a Linux instance using the console**

2. In the navigation pane, choose **Run Command**.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Run Command**.
3. Choose **Run command**.
4. In the **Command document** list, choose the `AWS-RunPowerShellScript` document.
5. In the **Command parameters** section, specify the available PowerShell commands you want to use.
6. In the **Targets** section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

   **Note**
   
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.
7. For **Other parameters**:
   - For **Comment**, type information about this command.
   - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.
8. (Optional) For **Rate control**:
   - For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

   **Note**
   
   If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
   - For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.
9. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.

   **Note**
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

   For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

11. Choose **Run**.

To see examples that use the `aws:runPowerShellScript` plugin, see `aws:runPowerShellScript` (p. 1074).

## Running commands using the document version parameter

You can use the document version parameter to specify which version of an SSM document to use when the command runs. You can specify one of the following options for this parameter:

- `$DEFAULT`
- `$LATEST`
- Version number

Use the following procedure to run a command using the document version parameter.

### Linux

**To run commands using the AWS CLI on Linux instances**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. List all available documents
   
   This command lists all of the documents available for your account based on IAM permissions.

   ```
   aws ssm list-documents
   ```

3. Use the following command to view the different versions of a document.

   ```
   aws ssm list-document-versions \\
   --name "document-name"
   ```

4. Use the following command to run a command that uses an SSM document version.

   ```
   aws ssm send-command \\
   --document-name "AWS-RunShellScript" \\
   --parameters commands="echo Hello" \\
   --instance-ids instance-ID \\
   --document-version '$LATEST'
   ```
Windows

**To run commands using the AWS CLI on Windows Server instances**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. List all available documents
   
   This command lists all of the documents available for your account based on IAM permissions.

   ```
   aws ssm list-documents
   ```

3. Use the following command to view the different versions of a document.

   ```
   aws ssm list-document-versions ^
   --name "document-name"
   ```

4. Use the following command to run a command that uses an SSM document version.

   ```
   aws ssm send-command ^
   --document-name "AWS-RunShellScript" ^
   --parameters commands="echo Hello" ^
   --instance-ids instance-ID ^
   --document-version "$LATEST"
   ```

PowerShell

**To run commands using the Tools for PowerShell**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. List all available documents
   
   This command lists all of the documents available for your account based on IAM permissions. The command returns a list of Linux and Windows documents.

   ```
   Get-SSMDocumentList
   ```

3. Use the following command to view the different versions of a document.

   ```
   Get-SSMDocumentVersionList `^`-Name "document-name"
   ```

4. Use the following command to run a command that uses an SSM document version.

   ```
   Send-SSMCommand `^`-DocumentName "AWS-RunShellScript" `^`
   -Parameter @{commands = "echo helloWorld"} `^`
   -InstanceIds "instance-ID" `^`
   -DocumentVersion "$LATEST"
   ```
Using targets and rate controls to send commands to a fleet

You can send commands to tens, hundreds, or thousands of instances by using the `targets` parameter (the **Specify instance tags** option in the **Run a command** page in the console). The `targets` parameter accepts a `Key, Value` combination based on Amazon EC2 tags that you specified for your instances. When you run the command, the system locates and attempts to run the command on all instances that match the specified tags. For more information about Amazon EC2 tags, see [Tagging your Amazon EC2 resources](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/tagging.html) in the *Amazon EC2 User Guide* (content applies to Windows Server and Linux instances). You can also send commands to instances that belong to an AWS resource group. For more information about resource groups, see [What are Resource Groups?](https://docs.aws.amazon.com/aws-resource-groups/latest/userguide/resource-groups-getting-started.html) in the *AWS Resource Groups User Guide*.

**Note**
You can also use the `targets` parameter to target a list of specific instance IDs, as described in the next section.

To control command execution across hundreds or thousands of instances, Run Command also includes parameters for restricting how many instances can simultaneously process a request and how many errors can be thrown by a command before the command is terminated.

### Contents
- Targeting multiple instances (p. 840)
- Using rate controls (p. 842)

### Targeting multiple instances

You can run a command and target instances by specifying tags applied to managed instances, AWS resource group names, or instance IDs.

**Note**
Sample commands in this section are truncated using ` [...]`. For use with the AWS CLI `send-command` command, the `targets` parameter supports the syntax demonstrated in the following examples:

#### Example 1: Targeting Tags

```
aws ssm send-command --document-name document-name --targets Key=tag:tag-name,Values=tag-value [...]
```

#### Example 2: Targeting an AWS Resource Group

You can specify a maximum of one resource group name per command. When you create a resource group, we recommend including `AWS::SSM:ManagedInstance` and `AWS::EC2::Instance` as resource types in your grouping criteria.

**Note**
In order to send commands that target a resource group, you must have been granted IAM permissions to list, or view, the resources that belong to that group. For more information, see [Set Up Permissions](https://docs.aws.amazon.com/aws-resource-groups/latest/userguide/resource-groups-getting-started.html) in the *AWS Resource Groups User Guide*.

```
aws ssm send-command --document-name --targets Key=resource-groups:name,Values=resource-group-name [...]
```

#### Example 3: Targeting Instance IDs

```
aws ssm send-command --document-name document-name --targets Key=instanceids,Values=instance-ID-1,instance-ID-2,instance-ID-3 [...]
```
If you tagged instances for different environments using a Key named Environment and Values of Development, Test, Pre-production and Production, then you could send a command to all of the instances in one of these environments by using the targets parameter with the following syntax:

```
aws ssm send-command --document-name document-name --targets
  Key=tag:Environment,Values=Development [...] 
```

You could target additional instances in other environments by adding to the Values list. Separate items using commas.

```
aws ssm send-command --document-name document-name --targets
  Key=tag:Environment,Values=Development,Test,Pre-production [...] 
```

Example: Refining your targets using multiple Key criteria

You can refine the number of targets for your command by including multiple Key criteria. If you include more than one Key criteria, the system targets instances that meet all of the criteria. The following command targets all instances tagged for the Finance Department and tagged for the database server role.

```
aws ssm send-command --document-name document-name --targets
  Key=tag:Department,Values=Finance
  Key=tag:ServerRole,Values=Database [...] 
```

Example: Using multiple Key and Value criteria

Expanding on the previous example, you can target multiple departments and multiple server roles by including additional items in the Values criteria.

```
aws ssm send-command --document-name document-name --targets
  Key=tag:Department,Values=Finance,Marketing
  Key=tag:ServerRole,Values=WebServer,Database [...] 
```

Example: Targeting tagged instances using multipleValues criteria

If you tagged instances for different environments using a Key named Department and Values of Sales and Finance, then you could send a command to all of the instances in these environments by using the targets parameter with the following syntax:

```
aws ssm send-command --document-name document-name --targets
  Key=tag:Department,Values=Sales,Finance [...] 
```

**Note**

You can specify a maximum of 5 keys, and 5 values for each key.

If either a tag key (the tag name) or a tag value includes spaces, then you must enclose the tag key or the value in quotation marks, as show in the following examples.

**Example 1: Spaces in Value tag.**

```
aws ssm send-command --document-name document-name --targets Key=tag:OS,Values="Windows Server 2016 Nano" [...] 
```

**Example 2: Spaces in tag key and Value.**
Running commands

Example 3: Spaces in one item in a list of Values.

```bash
aws ssm send-command --document-name document-name --targets
  Key=tag:Department,Values="Sales","Finance","Systems Mgmt" [...] 
```

Using rate controls

You can control the rate at which commands are sent to instances in a group by using concurrency controls and error controls.

Topics

- Using concurrency controls (p. 842)
- Using error controls (p. 842)

Using concurrency controls

You can control how many servers run the command at the same time by using the max-concurrency parameter (the Concurrency options in the Run a command page). You can specify either an absolute number of instances, for example 10, or a percentage of the target set, for example 10%. The queueing system delivers the command to a single instance and waits until the initial invocation completes before sending the command to two more instances. The system exponentially sends commands to more instances until the value of max-concurrency is met. The default for value max-concurrency is 50. The following examples show you how to specify values for the max-concurrency parameter:

```bash
aws ssm send-command --document-name document-name --max-concurrency 10 --targets
  Key=tag:Environment,Values=Development [...] 
```

```bash
aws ssm send-command --document-name document-name --max-concurrency 10% --targets
  Key=tag:Department,Values=Finance,Marketing
  Key=tag:ServerRole,Values=WebServer,Database [...] 
```

Using error controls

You can also control the execution of a command to hundreds or thousands of instances by setting an error limit using the max-errors parameters (the Error threshold field in the Run a command page). The parameter specifies how many errors are allowed before the system stops sending the command to additional instances. You can specify either an absolute number of errors, for example 10, or a percentage of the target set, for example 10%. If you specify 3, for example, the system stops sending the command when the fourth error is received. If you specify 0, then the system stops sending the command to additional instances after the first error result is returned. If you send a command to 50 instances and set max-errors to 10%, then the system stops sending the command to additional instances when the sixth error is received.

Invocations that are already running a command when max-errors is reached are allowed to complete, but some of these invocations may fail as well. If you need to ensure that there won't be more than max-errors failed invocations, set max-concurrency to 1 so the invocations proceed one at a time. The default for max-errors is 0. The following examples show you how to specify values for the max-errors parameter:

```bash
aws ssm send-command --document-name document-name --max-errors 10 --targets
  Key=tag:Database,Values=Development [...] 
```
Canceling a command

You can attempt to cancel a command as long as the service shows that it is in either a Pending or Executing state. However, even if a command is still in one of these states, we cannot guarantee that the command will be terminated and the underlying process stopped.

To cancel a command using the console

2. In the navigation pane, choose Run Command.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon to open the navigation pane, and then choose Run Command.
3. Select the command invocation that you want to cancel.
4. Choose Cancel command.

To cancel a command using the AWS CLI

Use the following command.

```
aws ssm cancel-command --command-id "command-ID" --instance-ids "instance-ID"
```

For information about the status of a cancelled command, see Understanding command statuses (p. 845).

Handling exit codes with scripts

In some cases, you might need to manage how exit codes are handled in your commands with the use of scripts.

Topics

- Rebooting managed instance from scripts (p. 843)
- Managing exit codes in Run Command commands (p. 844)

Rebooting managed instance from scripts

If the scripts that you run by using Run Command reboot managed instances, then you must specify an exit code in your script. If you attempt to reboot an instance from a script by using some other mechanism, the script execution status might not be updated correctly, even if the reboot is the last step in your script. For Windows managed instances, you specify `exit 3010` in your script. For Linux managed instances, you specify `exit 194`. The exit code instructs SSM Agent to reboot the managed instance, and then restart the script after the reboot completed. Before starting the reboot, SSM Agent informs the Systems Manager service in the cloud that communication will be disrupted during the server reboot.
Create idempotent scripts

When developing scripts that reboot managed instances, make the scripts idempotent so the script execution continues where it left off after the reboot. Idempotent scripts manage state and validate if the action was performed or not. This prevents a step from running multiple times when it is only intended to run once.

Here is an outline example of an idempotent script the reboots the instance multiple times.

```bash
$name = Get current computer name
If ($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
}
```

The following script samples use exit codes to restart instances. The Linux example installs package updates on Amazon Linux, and then restarts the instance. The Windows example installs the Hyper-V application on the instance, and then restarts the instance.

Amazon Linux example

```bash
#!/bin/bash
yum -y update
needs-restarting -r
if [ $? -eq 1 ]
then
    exit 194
else
    exit 0
fi
```

Windows example

```bash
#telnet = Get-WindowsFeature -Name Telnet-Client
if (-not #telnet.Installed)
{
    # Install Telnet and then send a reboot request to SSM Agent.
    Install-WindowsFeature -Name "Telnet-Client"
    exit 3010
}
```

Managing exit codes in Run Command commands

Run Command lets you define how exit codes are handled in your scripts. By default, the exit code of the last command run in a script is reported as the exit code for the entire script. For example, you have a
script that contains three commands. The first one fails but the following ones succeed. Because the final command succeeded, the status of the execution is reported as succeeded.

**Shell scripts**

To fail the entire command at the first failure, you can include a shell conditional statement to exit the script if any command before the final one fails. Use the following approach.

```
<command 1>
if [ $? != 0 ]
then
  exit <N>
fi
<command 2>
<command 3>
```

In the following example, the entire script fails if the first command fails.

```
cd /test
if [ $? != 0 ]
then
  echo "Failed"
  exit 1
fi
date
```

**PowerShell scripts**

PowerShell requires that you call exit explicitly in your scripts for Run Command to successfully capture the exit code.

```
<command 1>
if ($?) {<do something>}
else {exit <N>}
<command 2>
<command 3>
exit <N>
```

Here is an example:

```
cd C:\
if ($?) {echo "Success"}
else {exit 1}
date
```

**Understanding command statuses**

Systems Manager Run Command reports detailed status information about the different states a command experiences during processing and for each instance that processed the command. You can monitor command statuses using the following methods.

- Click the Refresh icon on the Run Command page in the Amazon EC2 console.
- Call `list-commands` or `list-command-invocations` using the AWS CLI. Or call `Get-SSMCommand` or `Get-SSMCommandInvocation` using AWS Tools for Windows PowerShell.
- Configure CloudWatch Events to log status changes.
- Configure Amazon SNS to send notifications for all status changes or specific statuses like Failed or TimedOut.
Run Command status

Run Command reports status details for three areas: plugins, invocations, and an overall command status. A plugin is a code-execution block that is defined in your command's Systems Manager (SSM) document. For more information about plugins, see SSM document plugin reference (p. 1052).

When you send a command to multiple instances at the same time, each copy of the command targeting each instance is a command invocation. For example, if you use the AWS-RunShellScript document and send an ifconfig command to 20 instances, that command has 20 invocations. Each command invocation individually reports status. The plugins for a given command invocation individually report status as well.

Lastly, Run Command includes an aggregated command status for all plugins and invocations. The aggregated command status can be different than the status reported by plugins or invocations, as noted in the following tables.

**Note**

If you run commands to large numbers of instances using the `max-concurrency` or `max-errors` parameters, command status reflects the limits imposed by those parameters, as described in the following tables. For more information about these parameters, see Using targets and rate controls to send commands to a fleet (p. 840).

### Detailed status for command plugins and invocations

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pending</td>
<td>The command was not yet received by the agent on the instance. If the command is not received by the agent before the value specified by the <strong>Timeout (seconds)</strong> parameter is reached, then the status changes to <strong>Delivery Timed Out</strong>.</td>
</tr>
<tr>
<td>InProgress</td>
<td>The command was received by the agent, or the command started executing on the instance. Depending on the result of all command plugins, the status will change to <strong>Success</strong>, <strong>Failed</strong>, or <strong>Execution Timed Out</strong>. If the agent is not available on the instance, the command status will show <strong>InProgress</strong> until the agent is available again. The status will then change to a terminal state.</td>
</tr>
<tr>
<td>Delayed</td>
<td>The system attempted to send the command to the instance but was not successful. The system will retry again.</td>
</tr>
</tbody>
</table>
| Success    | The command was received by SSM Agent on the instance and returned an exit code of zero. This status does not mean the command was successfully processed on the instance. This is a terminal state. **Note**  
To troubleshoot errors or get more information about the command execution, send a command that handles errors or exceptions by returning appropriate exit codes (non-zero exit codes for command failure). |
<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Timed Out</td>
<td>The command was not delivered to the instance before the delivery timeout expired. Delivery timeouts do not count against the parent command's max-errors limit, but they do contribute to whether the parent command status is Success or Incomplete. This is a terminal state.</td>
</tr>
<tr>
<td>Execution Timed Out</td>
<td>Command execution started on the instance, but the execution was not complete before the execution timeout expired. Execution timeouts count against the max-errors limit of the parent command. This is a terminal state. When the timeout is reached, Systems Manager stops the command execution.</td>
</tr>
<tr>
<td>Failed</td>
<td>The command was not successful on the instance. For a plugin, this indicates that the result code was not zero. For a command invocation, this indicates that the result code for one or more plugins was not zero. Invocation failures count against the max-errors limit of the parent command. This is a terminal state.</td>
</tr>
<tr>
<td>Canceled</td>
<td>The command was terminated before it was completed. This is a terminal state.</td>
</tr>
<tr>
<td>Undeliverable</td>
<td>The command can't be delivered to the instance. The instance might not exist or it might not be responding. Undeliverable invocations don't count against the parent command's max-errors limit, and they don't contribute to whether the parent command status is Success or Incomplete. This is a terminal state.</td>
</tr>
<tr>
<td>Terminated</td>
<td>The parent command exceeded its max-errors limit and subsequent command invocations were canceled by the system. This is a terminal state.</td>
</tr>
<tr>
<td>Invalid Platform</td>
<td>The command was sent to an instance that did not match the required platforms specified by the chosen document. InvalidPlatform does not count against the parent command's max-errors limit, and does not contribute to whether the parent command status is Success or Failed. This is a terminal state.</td>
</tr>
</tbody>
</table>
### Understanding command statuses

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Denied</td>
<td>The IAM user or role initiating the command does not have access to the targeted managed instance. <code>AccessDenied</code> does not count against the parent command's <code>max-errors</code> limit, but does contribute to whether the parent command status is <code>Success</code> or <code>Failed</code>. (For example, if all invocations in a command have the status <code>AccessDenied</code>, then the command status returned is <code>Failed</code>. However, if a command has 5 invocations, 4 of which return the status <code>AccessDenied</code> and 1 of which returns the status <code>Success</code>, then the parent command's status is <code>Success</code>.) This is a terminal state.</td>
</tr>
</tbody>
</table>

### Detailed status for a command

<table>
<thead>
<tr>
<th>Status</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pending</td>
<td>The command was not yet received by an agent on any instances.</td>
</tr>
<tr>
<td>InProgress</td>
<td>The command has been sent to at least one instance but has not reached a final state on all instances.</td>
</tr>
<tr>
<td>Delayed</td>
<td>The system attempted to send the command to the instance but was not successful. The system will retry again.</td>
</tr>
<tr>
<td>Success</td>
<td>The command was received by SSM Agent on all specified or targeted instances and returned an exit code of zero. All command invocations have reached a terminal state, and the value of <code>max-errors</code> was not reached. This status does not mean the command was successfully processed on all specified or targeted instances. This is a terminal state.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>To troubleshoot errors or get more information about the command execution, send a command that handles errors or exceptions by returning appropriate exit codes (non-zero exit codes for command failure).</td>
</tr>
<tr>
<td>Delivery Timed Out</td>
<td>The command was not delivered to the instance before the delivery timeout expired. The value of <code>max-errors</code> or more command invocations shows a status of <code>Delivery Timed Out</code>. This is a terminal state.</td>
</tr>
<tr>
<td>Execution Timed Out</td>
<td>Command execution started on the instance, but the execution was not complete before the execution timeout expired. The value of <code>max-errors</code> or more command invocations shows</td>
</tr>
<tr>
<td>Status</td>
<td>Details</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Failed</td>
<td>The command was not successful on the instance. The value of max-errors or more command invocations shows a status of Failed. This is a terminal state.</td>
</tr>
<tr>
<td>Incomplete</td>
<td>The command was attempted on all instances and one or more of the invocations does not have a value of Success. However, not enough invocations failed for the status to be Failed. This is a terminal state.</td>
</tr>
<tr>
<td>Canceled</td>
<td>The command was terminated before it was completed. This is a terminal state.</td>
</tr>
<tr>
<td>Rate Exceeded</td>
<td>The number of instances targeted by the command exceeded the account limit for pending invocations. The system has canceled the command before executing it on any instance. This is a terminal state.</td>
</tr>
<tr>
<td>Access Denied</td>
<td>The IAM user or role initiating the command does not have access to the targeted resource group. AccessDenied does not count against the parent command's max-errors limit, but does contribute to whether the parent command status is Success or Failed. (For example, if all invocations in a command have the status AccessDenied, then the command status returned is Failed. However, if a command has 5 invocations, 4 of which return the status AccessDenied and 1 of which returns the status Success, then the parent command's status is Success.) This is a terminal state.</td>
</tr>
<tr>
<td>No Instances In Tag</td>
<td>The tag key-pair value or resource group targeted by the command does not match any managed instances. This is a terminal state.</td>
</tr>
</tbody>
</table>

**Understanding command timeout values**

Systems Manager enforces the following timeout values when running commands.

**Delivery Timeout**

In the Systems Manager console, you specify the delivery timeout value in the Timeout (seconds) field. Systems Manager must deliver the command to the targets and the SSM Agent on the targets must begin processing the command within the number of seconds specified for Timeout (seconds).
Execution Timeout

In the Systems Manager console, you specify the execution timeout value in the Execution Timeout field, if available. Not all SSM documents require that you specify an execution timeout. If specified, the command must complete within this time period.

Default Execution Timeout

If a Systems Manager (SSM) document doesn't require that you explicitly specify an execution timeout value, then Systems Manager enforces the hard-coded default execution timeout.

Total Timeout

Total timeout is equal to the value of delivery timeout plus execution timeout. If execution timeout is not required by the SSM document, then total timeout is equal to the value of delivery timeout plus default execution timeout.

How Systems Manager Reports Timeouts

If Systems Manager receives an execution timeout reply from SSM Agent on a target, then Systems Manager marks the command invocation as executionTimeout.

If Systems Manager doesn't receive any reply from SSM Agent on a target, and the command execution exceeds the total timeout value, then Systems Manager marks the command invocation as deliveryTimeout.

To determine timeout status on a target, SSM Agent combines all of the parameters and the content of the SSM document to calculate for executionTimeout. When SSM Agent determines that a command has timed out, it sends executionTimeout to the service.

Note

SSM Agent processes executionTimeout differently depending on the type of SSM document and the document version.

Run Command walkthroughs

The walkthroughs in this section show you how to run commands with Run Command using either the AWS Command Line Interface or AWS Tools for Windows PowerShell.

Contents

- Walkthrough: Use the AWS CLI with Run Command (p. 850)
- Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command (p. 853)

You can also view sample commands in the following references.

- Systems Manager AWS CLI Reference
- Systems Manager AWS Tools for Windows PowerShell Reference

Walkthrough: Use the AWS CLI with Run Command

The following sample walkthrough shows you how to use the AWS CLI to view information about commands and command parameters, how to run commands, and how to view the status of those commands.
Important
Only trusted administrators should be allowed to use Systems Manager pre-configured documents shown in this topic. The commands or scripts specified in Systems Manager documents run with administrative privilege on your instances. If a user has permission to run any of the pre-defined Systems Manager documents (any document that begins with AWS), then that user also has administrator access to the instance. For all other users, you should create restrictive documents and share them with specific users. For more information about restricting access to Run Command, see Create non-Admin IAM users and groups for Systems Manager (p. 27).

Topics
• Step 1: Getting started (p. 851)
• Step 2: Run shell scripts (p. 852)
• Step 3: Send a command using the AWS-RunShellScript document - Example 1 (p. 852)
• Step 4: Send a command using the AWS-RunShellScript document - Example 2 (p. 852)
• Additional examples (p. 853)

Step 1: Getting started
You must either have administrator privileges on the instances you want to configure or you must have been granted the appropriate permission in IAM. Also note, this example uses the US East (Ohio) Region (us-east-2). Run Command is currently available in the AWS Regions listed in Systems Manager service endpoints in the Amazon Web Services General Reference. For more information, see Systems Manager prerequisites (p. 14).

To run commands using the AWS CLI
1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).
2. List all available documents
   This command lists all of the documents available for your account based on IAM permissions. The command returns a list of Linux and Windows documents.
   ```bash
   aws ssm list-documents
   ```
3. Verify that an instance is ready to receive commands
   The output of the following command shows if instances are online.
   ```bash
   aws ssm describe-instance-information --output text --query "InstanceInformationList[*]"
   ```
4. Use the following command to view details about a particular instance.
   **Note**
   To run the commands in this walkthrough, you must replace the instance and command IDs. The command ID is returned as a response of the send-command. The instance ID is available from the Amazon EC2 console.
   ```bash
   aws ssm describe-instance-information --instance-information-filter-list key=InstanceIds,valueSet=instance-ID
   ```
Step 2: Run shell scripts

Using Run Command and the AWS-RunShellScript document, you can run any command or script on an EC2 instance as if you were logged on locally.

To view the description and available parameters

- Use the following command to view a description of the Systems Manager JSON document.

  ```
  aws ssm describe-document --name "AWS-RunShellScript" --query "[Document.Name,Document.Description]"
  ```

- Use the following command to view the available parameters and details about those parameters.

  ```
  aws ssm describe-document --name "AWS-RunShellScript" --query "Document.Parameters[*]"
  ```

Step 3: Send a command using the AWS-RunShellScript document - Example 1

Use the following command to get IP information for an instance.

  ```
  aws ssm send-command --instance-ids "instance-ID" --document-name "AWS-RunShellScript" --comment "IP config" --parameters commands=ifconfig --output text
  ```

Get command information with response data

The following command uses the Command ID that was returned from the previous command to get the details and response data of the command execution. The system returns the response data if the command completed. If the command execution shows "Pending" or "InProgress" you will need to run this command again to see the response data.

  ```
  aws ssm list-command-invocations --command-id $sh_command_id --details
  ```

Step 4: Send a command using the AWS-RunShellScript document - Example 2

The following command displays the default user account running the commands.

  ```
  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")
  ```

Get command status

The following command uses the Command ID to get the status of the command execution on the instance. This example uses the Command ID that was returned in the previous command.

  ```
  aws ssm list-commands --command-id "$command-ID"
  ```

Get command details

The following command uses the Command ID from the previous command to get the status of the command execution on a per instance basis.

  ```
  aws ssm list-command-invocations --command-id "$command-ID" --details
  ```
Get command information with response data for a specific instance

The following command returns the output of the original `aws ssm send-command` for a specific instance.

```
aws ssm list-command-invocations --instance-id instance-ID --command-id "command-ID" --details
```

Additional examples

The following command returns the version of Python running on an instance.

```
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}"'
```

The following command runs a Python script using Run Command.

```
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}"'
```

Walkthrough: Use the AWS Tools for Windows PowerShell with Run Command

The following examples show how to use the Tools for Windows PowerShell to view information about commands and command parameters, how to run commands, and how to view the status of those commands. This walkthrough includes an example for each of the pre-defined Systems Manager documents.

**Important**

Only trusted administrators should be allowed to use Systems Manager pre-configured documents shown in this topic. The commands or scripts specified in Systems Manager documents run with administrative privilege on your instances. If a user has permission to run any of the pre-defined Systems Manager documents (any document that begins with AWS), then that user also has administrator access to the instance. For all other users, you should create restrictive documents and share them with specific users. For more information about restricting access to Run Command, see Create non-Admin IAM users and groups for Systems Manager (p. 27).

Topics

- Configure AWS Tools for Windows PowerShell session settings (p. 854)
- List all available documents (p. 854)
- Run PowerShell commands or scripts (p. 854)
- Install an application using the AWS-InstallApplication document (p. 855)
- Install a PowerShell module using the AWS-InstallPowerShellModule JSON document (p. 856)
- Join an instance to a Domain using the AWS-JoinDirectoryServiceDomain JSON document (p. 857)
- Send Windows metrics to amazon CloudWatch using the AWS-ConfigureCloudWatch document (p. 858)
- Update EC2Config using the AWS-UpdateEC2Config document (p. 859)
Configure AWS Tools for Windows PowerShell session settings

Specify your credentials

Open AWS Tools for Windows PowerShell on your local computer and run the following command to specify your credentials. You must either have administrator privileges on the instances you want to configure or you must have been granted the appropriate permission in IAM. For more information, see Systems Manager prerequisites (p. 14).

```
Set-AWSCredentials -AccessKey key-name -SecretKey key-name
```

Set a default AWS Region

Run the following command to set the region for your PowerShell session. The example uses the US East (Ohio) Region (us-east-2). Run Command is currently available in the AWS Regions listed in Systems Manager service endpoints in the Amazon Web Services General Reference.

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

List all available documents

This command lists all of the documents available for your account:

```
Get-SSMDocumentList
```

Run PowerShell commands or scripts

Using Run Command and the AWS-RunPowerShell document, you can run any command or script on an EC2 instance as if you were logged onto the instance using Remote Desktop. You can issue commands or type in a path to a local script to run the command.

**Note**

For information about rebooting servers and instances when using Run Command to call scripts, see Rebooting managed instance from scripts (p. 843).

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-RunPowerShellScript"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-RunPowerShellScript" | Select -ExpandProperty Parameters
```

Send a command using the AWS-RunPowerShellScript document

The following command shows the contents of the "C:\Users" directory and the contents of the "C:\" directory on two instances.
Get command request details

The following command uses the `CommandId` to get the status of the command execution on both instances. This example uses the `CommandId` that was returned in the previous command.

```
Get-SSMCommand -CommandId $runPSCommand.CommandId
```

The status of the command in this example can be Success, Pending, or InProgress.

Get command information per instance

The following command uses the `CommandId` from the previous command to get the status of the command execution on a per instance basis.

```
Get-SSMCommandInvocation -CommandId $runPSCommand.CommandId
```

Get command information with response data for a specific instance

The following command returns the output of the original `Send-SSMCommand` for a specific instance.

```
Get-SSMCommandInvocation `-CommandId $runPSCommand.CommandId `-Details $true `-InstanceId instance-ID` | Select -ExpandProperty CommandPlugins
```

Cancel a command

The following command cancels the `Send-SSMCommand` for the `AWS-RunPowerShellScript` document.

```
#cancelCommand = Send-SSMCommand `
-InstanceIds @("instance-ID-1","instance-ID-2") `
-DocumentName "AWS-RunPowerShellScript" `
-Comment "Demo AWS-RunPowerShellScript with two instances" `
-Parameter @{'commands'='Start-Sleep -Seconds 120; dir C:\'}
Stop-SSMCommand -CommandId #cancelCommand.CommandId
```

Check the command status

The following command checks the status of the `Cancel` command.

```
Get-SSMCommand -CommandId #cancelCommand.CommandId
```

Install an application using the `AWS-InstallApplication` document

Using Run Command and the `AWS-InstallApplication` document, you can install, repair, or uninstall applications on instances. The command requires the path or address to an MSI.
Note
For information about rebooting servers and instances when using Run Command to call scripts, see Rebooting managed instance from scripts (p. 843).

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallApplication"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallApplication" | Select -ExpandProperty Parameters
```

Send a command using the AWS-InstallApplication document

The following command installs a version of Python on your instance in unattended mode, and logs the output to a local text file on your C: drive.

```
#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'='noredirect /quiet /log c:\pythoninstall.txt'}
```

Get command information per instance

The following command uses the CommandId to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId $installAppCommand.CommandId -Details $true
```

Get command information with response data for a specific instance

The following command returns the results of the Python installation.

```
Get-SSMCommandInvocation ` 
-CommandId $installAppCommand.CommandId ` 
-Details $true ` 
-InstanceId instance-ID ` 
| Select -ExpandProperty CommandPlugins
```

Install a PowerShell module using the AWS-InstallPowerShellModule JSON document

You can use Run Command to install PowerShell modules on an EC2 instance. For more information about PowerShell modules, see Windows PowerShell Modules.

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallPowerShellModule"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-InstallPowerShellModule" | Select -ExpandProperty Parameters
```
Install a PowerShell module

The following command downloads the EZOut.zip file, installs it, and then runs an additional command to install XPS viewer. Lastly, the output of this command is uploaded to an S3 bucket named "demo-ssm-output-bucket".

```
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
```

Get command information per instance

The following command uses the CommandId to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId $installPSCommand.CommandId -Details $true
```

Get command information with response data for the instance

The following command returns the output of the original Send-SSMCommand for the specific CommandId.

```
Get-SSMCommandInvocation
  -CommandId $installPSCommand.CommandId
  -Details $true
| Select -ExpandProperty CommandPlugins
```

Join an instance to a Domain using the AWS-JoinDirectoryServiceDomain JSON document

Using Run Command, you can quickly join an instance to an AWS Directory Service domain. Before executing this command you must create a directory. We also recommend that you learn more about the AWS Directory Service. For more information, see What Is AWS Directory Service?.

Currently you can only join an instance to a domain. You cannot remove an instance from a domain.

**Note**

For information about rebooting servers and instances when using Run Command to call scripts, see Rebooting managed instance from scripts (p. 843).

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-JoinDirectoryServiceDomain"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-JoinDirectoryServiceDomain" | Select -ExpandProperty Parameters
```

Join an instance to a domain

The following command joins the instance to the given AWS Directory Service domain and uploads any generated output to the example S3 bucket.
#domainJoinCommand = Send-SSMCommand ` 
-InstanceId instance-ID` 
-DocumentName "AWS-JoinDirectoryServiceDomain" ` 
-Parameter @{"directoryId"='d-example01'; 'directoryName'='ssm.example.com'; 'dnsIpAddresses'=('@(192.168.10.195', '192.168.20.97')} } ` 
-OutputS3BucketName demo-ssm-output-bucket

## Get command information per instance

The following command uses the CommandId to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId #domainJoinCommand.CommandId -Details $true
```

## Get command information with response data for the instance

This command returns the output of the original Send-SSMCommand for the specific CommandId.

```
Get-SSMCommandInvocation ` 
-CommandId #domainJoinCommand.CommandId ` 
-Details $true ` 
| Select -ExpandProperty CommandPlugins
```

## Send Windows metrics to amazon CloudWatch using the AWS-ConfigureCloudWatch document

You can send Windows Server messages in the application, system, security, and Event Tracing for Windows (ETW) logs to Amazon CloudWatch Logs. When you enable logging for the first time, Systems Manager sends all logs generated within one (1) minute from the time that you start uploading logs for the application, system, security, and ETW logs. Logs that occurred before this time are not included. If you disable logging and then later re-enable logging, Systems Manager sends logs from the time it left off. For any custom log files and Internet Information Services (IIS) logs, Systems Manager reads the log files from the beginning. In addition, Systems Manager can also send performance counter data to Amazon CloudWatch.

If you previously enabled CloudWatch integration in EC2Config, the Systems Manager settings override any settings stored locally on the instance in the C:\Program Files\Amazon\EC2ConfigService\Settings\AWS.EC2.Windows.CloudWatch.json file. For more information about using EC2Config to manage performance counters and logs on single instance, see Sending Performance Counters to CloudWatch and Logs to CloudWatch Logs.

## View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch"
```

## View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-ConfigureCloudWatch" | Select -ExpandProperty Parameters
```

## Send application logs to CloudWatch

The following command configures the instance and moves Windows Applications logs to CloudWatch.

```
#cloudWatchCommand = Send-SSMCommand ` 
"domainJoinCommand"
```
Get command information per instance

The following command uses the CommandId to get the status of the command execution.

```
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true
```

Get command information with response data for a specific instance

The following command returns the results of the Amazon CloudWatch configuration.

```
Get-SSMCommandInvocation -CommandId $cloudWatchCommand.CommandId -Details $true -InstanceId instance-ID | Select -ExpandProperty CommandPlugins
```

Send performance counters to CloudWatch using the AWS-ConfigureCloudWatch document

The following demonstration command uploads performance counters to CloudWatch. For more information, see the `Amazon CloudWatch User Guide`

```
#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,AWS.EC2.Windows.CloudWatch', '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': 'region', 'NameSpace': 'Windows-Default'}, 'Flows': ['PerformanceCounter,CloudWatch']}}}})
```

Update EC2Config using the AWS-UpdateEC2Config document

Using Run Command and the AWS-EC2ConfigUpdate document, you can update the EC2Config service running on your Windows Server instances. This command can update the EC2Config service to the latest version or a version you specify.

View the description and available parameters

```
Get-SSMDocumentDescription -Name "AWS-UpdateEC2Config"
```

View more information about parameters

```
Get-SSMDocumentDescription -Name "AWS-UpdateEC2Config" | Select -ExpandProperty Parameters
```
Update EC2Config to the latest version

``` powershell
$ec2ConfigCommand = Send-SSMCommand -InstanceId instance-ID -DocumentName "AWS-UpdateEC2Config"
```

Get command information with response data for the instance

This command returns the output of the specified command from the previous `Send-SSMCommand`:

``` powershell
Get-SSMCommandInvocation `-CommandId $ec2ConfigCommand.CommandId` `-Details $true` `-InstanceId instance-ID` | Select -ExpandProperty CommandPlugins
```

Update EC2Config to a specific version

The following command will downgrade EC2Config to an older version:

``` powershell
Send-SSMCommand `-InstanceId instance-ID` `-DocumentName "AWS-UpdateEC2Config"` `-Parameter @{'version'='4.9.3519'; 'allowDowngrade'='true'}
```

Enable/Disable Windows automatic update using the AWS-ConfigureWindowsUpdate document

Using Run Command and the AWS-ConfigureWindowsUpdate document, you can enable or disable automatic Windows updates on your Windows Server instances. This command configures the Windows update agent to download and install Windows updates on the day and hour that you specify. If an update requires a reboot, the computer reboots automatically 15 minutes after updates have been installed. With this command you can also configure Windows update to check for updates but not install them. The AWS-ConfigureWindowsUpdate document is compatible with Windows Server 2008, 2008 R2, 2012, 2012 R2, and 2016.

View the description and available parameters

``` powershell
Get-SSMDocumentDescription -Name "AWS-ConfigureWindowsUpdate"
```

View more information about parameters

``` powershell
Get-SSMDocumentDescription -Name "AWS-ConfigureWindowsUpdate" | Select -ExpandProperty Parameters
```

Enable Windows automatic update

The following command configures Windows Update to automatically download and install updates daily at 10:00 pm.

``` powershell
#configureWindowsUpdateCommand = Send-SSMCommand `-InstanceId instance-ID` `-DocumentName "AWS-ConfigureWindowsUpdate"` `-Parameters @{'updateLevel'='InstallUpdatesAutomatically'; 'scheduledInstallDay'='Daily'; 'scheduledInstallTime'='22:00'}
```
View command status for enabling Windows automatic update

The following command uses the CommandId to get the status of the command execution for enabling Windows Automatic Update.

```
Get-SSMCommandInvocation `  
-Details $true `  
-CommandId $configureWindowsUpdateCommand.CommandId `  
| Select -ExpandProperty CommandPlugins
```

Disable Windows automatic update

The following command lowers the Windows Update notification level so the system checks for updates but does not automatically update the instance.

```
$configureWindowsUpdateCommand = Send-SSMCommand `  
-InstanceId instance-ID `  
-DocumentName "AWS-ConfigureWindowsUpdate" `  
-Parameters @{'updateLevel'='NeverCheckForUpdates'}
```

View command status for disabling Windows automatic update

The following command uses the CommandId to get the status of the command execution for disabling Windows automatic update.

```
Get-SSMCommandInvocation `  
-Details $true `  
-CommandId $configureWindowsUpdateCommand.CommandId `  
| Select -ExpandProperty CommandPlugins
```

Manage Windows updates using Run Command

Using Run Command and the AWS-InstallWindowsUpdates document, you can manage updates for EC2 instances for Windows Server. This command scans for or installs missing updates on your EC2 instances for Windows Server and optionally reboots following installation. You can also specify the appropriate classifications and severity levels for updates to install in your environment.

**Note**

For information about rebooting servers and instances when using Run Command to call scripts, see [Rebooting managed instance from scripts (p. 843)](#).

The following examples demonstrate how to perform the specified Windows Update management tasks.

**Search for all missing Windows updates**

```
Send-SSMCommand `  
-InstanceId instance-ID `  
-DocumentName "AWS-InstallWindowsUpdates" `  
-Parameters @{'Action'='Scan'}
```

**Install specific Windows updates**

```
Send-SSMCommand `  
-InstanceId instance-ID `  
-DocumentName "AWS-InstallWindowsUpdates" `
Install important missing Windows updates

Send-SSMCommand
-InstanceId instance-ID
-DocumentName "AWS-InstallWindowsUpdates"
-Parameters @{'Action'='Install';'SeverityLevels'='Important';'AllowReboot'='True'}

Install missing Windows updates with specific exclusions

Send-SSMCommand
-InstanceId instance-ID
-DocumentName "AWS-InstallWindowsUpdates"
-Parameters @{'Action'='Install';'ExcludeKbs'='kb-ID-1,kb-ID-2';'AllowReboot'='True'}

Troubleshooting Systems Manager Run Command

Run Command provides status details with each command execution. For more information about the details of command statuses, see Understanding command statuses (p. 845). You can also use the information in this topic to help troubleshoot problems with Run Command.

Topics
- Where are my instances? (p. 862)
- A step in my script failed, but the overall status is 'succeeded' (p. 863)
- What's the status of my Windows instances? (p. 863)
- What's the status of my Linux instances? (p. 863)
- Troubleshooting SSM Agent (p. 864)

Where are my instances?

In the Run a command page, after you choose an SSM document to run and select Manually selecting instances in the Targets section, a list is displayed of instances you can choose to run the command on.

After you create, activate, reboot, or restart a managed instance, install SSM Agent on an instance, or attach an IAM instance profile to an instance, it can take a few minutes for the instance to appear in the list.

If an instance you expect to see is still not listed, check the following requirements.

- **SSM Agent**: Make sure the latest version of SSM Agent is installed on the instance. Only Amazon Machine Images (AMIs) for Windows Server and some Linux AMIs are pre-configured with SSM Agent. For information about installing or reinstalling SSM Agent on an instance, see Installing and configuring SSM Agent on EC2 instances for Linux (p. 69) or Installing and configuring SSM Agent on Windows Server instances (p. 65).
- **IAM instance role**: Verify that the instance is configured with an AWS Identity and Access Management (IAM) role that enables the instance to communicate with the Systems Manager API. Also verify that your user account has an IAM user trust policy that enables your account to communicate with the Systems Manager API. For more information, see Create an IAM instance profile for Systems Manager (p. 30).
- **Service Endpoint connectivity**: Verify that the instance has connectivity to the Systems Manager service endpoints. This connectivity is provided by creating and configuring VPC endpoints for Systems Manager (p. 30).
Manager, or by allowing HTTPS (port 443) outbound traffic to the service endpoints. For more information, see Step 6: (Optional) Create a Virtual Private Cloud endpoint (p. 37).

- **Target operating system type:** Double-check that you have selected an SSM document that supports the type of instance you want to update. Most SSM documents support both Windows and Linux instances, but some do not. For example, if you select the SSM document AWS-InstallPowerShellModule, which applies only to Windows Server instances, you will not see Linux instances in the target instances list.

**A step in my script failed, but the overall status is 'succeeded'**

Run Command lets you define how exit codes are handled in your scripts. By default, the exit code of the last command run in a script is reported as the exit code for the entire script. You can, however, include a conditional statement to exit the script if any command before the final one fails. For information and examples, see Managing exit codes in Run Command commands (p. 844).

**What's the status of my Windows instances?**

Use the following PowerShell command to get status details about one or more instances:

```
Get-SSMInstanceInformation -InstanceInformationFilterList @{Key="InstanceIds";ValueSet="instance-ID-1","instance-ID-2"}
```

Use the following PowerShell command with no filters to see all instances registered to your account that are currently reporting an online status. Substitute the ValueSet="Online" with "ConnectionLost" or "Inactive" to view those statuses:

```
Get-SSMInstanceInformation -InstanceInformationFilterList @{Key="PingStatus";ValueSet="Online"}
```

Use the following command to see which instances are running the latest version of the EC2Config service. Substitute ValueSet="LATEST" with a specific version (for example, 3.0.54 or 3.10) to view those details:

```
Get-SSMInstanceInformation -InstanceInformationFilterList @{Key="AgentVersion";ValueSet="LATEST"}
```

**What's the status of my Linux instances?**

Use the following AWS CLI command to get status details about one or more instances:

```
aws ssm describe-instance-information --instance-information-filter-list key=InstanceIds,valueSet=instance-ID
```

Use the following command with no filters to see all instances registered to your account that are currently reporting an online status. Substitute the ValueSet="Online" with "ConnectionLost" or "Inactive" to view those statuses:

```
aws ssm describe-instance-information --instance-information-filter-list key=PingStatus,valueSet=Online
```

Use the following command to see which instances are running the latest version of SSM Agent. Substitute ValueSet="LATEST" with a specific version (for example, 1.0.145 or 1.0) to view those details:
If the describe-instance-information API operation returns an AgentStatus of Online, then your instance is ready to be managed using Run Command. If the status is Inactive, the instance has one or more of the following problems.

- SSM Agent is not installed.
- The instance does not have outbound internet connectivity.
- The instance was not launched with an IAM role that enables it to communicate with the SSM API, or the permissions for the IAM role are not correct for Run Command. For more information, see Create an IAM instance profile for Systems Manager (p. 30).

### Troubleshooting SSM Agent

If you experience problems executing commands using Run Command, there might be a problem with SSM Agent. Use the following information to help you view SSM Agent log files and troubleshoot the agent.

**Topics**

- View SSM Agent log files (p. 864)

### View SSM Agent log files

SSM Agent logs information in the following files. The information in these files can help you troubleshoot problems.

**Note**

If you choose to view these logs by using Windows File Explorer, be sure to enable the viewing of hidden files and system files in Folder Options.

**On Windows**

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

**On Linux**

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

### AWS Systems Manager State Manager

AWS Systems Manager State Manager is a secure and scalable configuration management service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define.

The following list describes the types of tasks you can perform with State Manager.

- Bootstrap instances with specific software at start-up
- Download and update agents on a defined schedule, including SSM Agent
- Configure network settings
• Join instances to a Windows domain (Windows Server instances only).
• Patch instances with software updates throughout their lifecycle
• Run scripts on Linux and Windows managed instances throughout their lifecycle

State Manager integrates with AWS CloudTrail to provide a record of all executions that you can audit, and Amazon CloudWatch Events to track state changes. You can also choose to store and view detailed command output in Amazon S3. For more information, see the following topics:

- Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144)
- Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147)
- (Optional) Set Up integrations with other AWS services (p. 42)

**Getting started with State Manager**

Complete the following tasks to get started with State Manager.

<table>
<thead>
<tr>
<th>Task</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Systems Manager prerequisites</td>
<td>Systems Manager prerequisites (p. 14)</td>
</tr>
<tr>
<td>Learn more about State Manager</td>
<td>About State Manager (p. 865)</td>
</tr>
<tr>
<td>Create and assign a State Manager association to your instances</td>
<td>Create an association (p. 870)</td>
</tr>
</tbody>
</table>

**Related content**

See the following blog posts for additional examples of how to use State Manager:

- Combating Configuration Drift Using Amazon EC2 Systems Manager and Windows PowerShell DSC
- Configure Amazon EC2 Instances in an Auto Scaling Group Using State Manager

**Topics**

- About State Manager (p. 865)
- Working with associations in Systems Manager (p. 867)
- AWS Systems Manager State Manager walkthroughs (p. 887)

**About State Manager**

AWS Systems Manager State Manager is a secure and scalable service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define.

Here’s how it works:

1. **Determine the state you want to apply to your managed instances.**

   Do you want to ensure that your managed instance are configured with specific applications, such as anti-virus or malware applications? Do you want to automate the process of updating the SSM Agent or other AWS packages such as AWSPVDriver? Do you need to ensure that specific ports are closed or open? To get started with State Manager, determine the state that you want to apply to your managed instances. The state that you want to apply will determine which SSM document you use to create a State Manager association.
A State Manager association is a configuration that is assigned to your managed instances. The configuration defines the state that you want to maintain on your instances. For example, an association can specify that anti-virus software must be installed and running on your instances, or that certain ports must be closed. The association specifies a schedule for when the configuration is reapplied. The association also specifies actions to take when applying the configuration. For example, an association for anti-virus software might run once a day. If the software is not installed, then State Manager installs it. If the software is installed, but the service is not running, then the association might instruct State Manager to start the service.

2. **Determine if a preconfigured SSM document can help you create the State Manager association.**

State Manager uses SSM documents to create an association. Systems Manager includes dozens of preconfigured SSM documents that you can use to create an association. Preconfigured documents are ready to perform common tasks like installing applications, configuring Amazon CloudWatch, running Systems Manager Automations, running PowerShell and Shell scripts, and joining a Directory Service domain for Active Directory, to name a few. You simply need to specify the name of the document and information for the required parameters, and then run the command to create the association. You can view all SSM documents in the Systems Manager console.

You can then choose the name of a document to learn more about each one. Here are two examples: AWS-ConfigureAWSPackage and AWS-InstallApplication.

3. **Create the association.**

You can create the association by using the AWS Systems Manager console, the AWS CLI, AWS Tools for Windows PowerShell, or the Systems Manager API. When you create the association, you specify the following information:

1. The parameters for the SSM document (for example, the path to the application to install or the script to run on the instances).
2. A schedule for when or how often to apply the state. You can specify a cron or rate expression. For more information about creating schedules by using cron and rate expressions, see Cron and rate expressions for associations (p. 1193).
3. Targets for the association. You can target managed instances by specifying Amazon EC2 tags, by choosing individual instance IDs, or by choosing a group in AWS Resource Groups. You can also target all managed instances in the current AWS Region and AWS account.

When you run the command to create the association, Systems Manager binds the information you specified (schedule, targets, SSM document, and parameters) to the managed instances. The status of the association initially shows "Pending" as the system attempts to reach all targets and immediately apply the state specified in the association.

**Note**

If you create a new association that is scheduled to run while an earlier association is still running, the earlier association is timed out and the new association runs.

Systems Manager reports the status of the request to create associations on the managed instances. You can view status details in the console or by using the DescribeInstanceAssociationsStatus API action. If you choose to write the output of the command to Amazon S3 when you create an association, you can also view the output in the Amazon S3 bucket you specify.

For more information, see Create an association (p. 870).

4. **Monitor and update.**

After you create the association, State Manager reapplies the configuration according to the schedule that you defined in the association. You can view the status of your associations on the State Manager page in the console or by directly calling the association ID generated by Systems Manager when you created the association. For more information, see Viewing association histories (p. 880). You can update your association documents and reapply them as necessary. You can also create multiple versions of an association. For more information, see Edit and create a new version of an association (p. 876).
Working with associations in Systems Manager

A State Manager association is a configuration that is assigned to your managed instances. The configuration defines the state that you want to maintain on your instances. For example, an association can specify that anti-virus software must be installed and running on your instances, or that certain ports must be closed. The association specifies a schedule for when the configuration is reapplied. The association also specifies actions to take when applying the configuration. For example, an association for anti-virus software might run once a day. If the software is not installed, then State Manager installs it. If the software is installed, but the service is not running, then the association might instruct State Manager to start the service.

Use the following topics to help you create and manage State Manager associations.

**Topics**
- About targets and rate controls in State Manager associations (p. 867)
- Create an association (p. 870)
- Edit and create a new version of an association (p. 876)
- Viewing association histories (p. 880)

**About targets and rate controls in State Manager associations**

This topic describes State Manager features that help you deploy an association to dozens or hundreds of instances while controlling how many instances run the association at the scheduled time.

**Targets**

When you create a State Manager association, you choose which instances to configure with the association in the **Targets** section of the Systems Manager console, as shown here.

If you create an association by using a command line tool such as the AWS CLI, then you specify the `targets` parameter. Targeting instances lets you configure tens, hundreds, or thousands of instances with an association without having to specify or choose individual instance IDs.

**Note**

You can only target Systems Manager managed instances. This means you must set up and configure your instances for Systems Manager before you create an association. For more information, see Setting up AWS Systems Manager (p. 25).
State Manager includes the following target options when creating an association.

Specify instance tags

Use this option to specify a tag key and (optionally) a tag value that are currently assigned to your instances. When you run the request, the system locates and attempts to create the association on all instances that match the specified tag key and value. When the system initially creates the association, it runs the association. After this initial run, the system runs the association according to the schedule you specified.

If you create new instances and assign the specified tag key and value to those instances, the system automatically applies the association, runs it immediately, and then runs it according to the schedule. If you delete the specified tags from an instance, the system no longer runs the association on those instances.

For information about assigning tags to your instances, see Tagging Your Amazon EC2 Resources in the Amazon EC2 User Guide.

Choose instances manually

Use this option to manually select the instances where you want to create the association. The Instances pane displays all Systems Manager managed instances in the current AWS account and Region. You can manually select as many instances as you want. If you don't see instances in the list for which you want to create an association, verify that the instances are running and are configured for Systems Manager. When the system initially creates the association, it runs the association. After this initial run, the system runs the association according to the schedule you specified.

Choose a resource group

Use this option to create an association on all instances returned by an AWS Resource Groups tag-based or AWS CloudFormation stack-based query.

Note the following details about targeting resource groups for an association.

- If you add new instances to a group, the system automatically maps the instances to the association that targets the resource group. The system applies the association to the instances when it discovers the change. After this initial run, the system runs the association according to the schedule you specified.
- If you delete a resource group, all instances in that group no longer run the association. As a best practice, you should delete associations targeting the group.
- You can only target a single resource group for an association. Multiple or nested groups are not supported.
- After you create an association, State Manager periodically updates the association with information about resources in the Resource Group. If you add new resources to a Resource Group, the schedule for when the system applies the association to the new resources depends on several factors. You can check the status of the association in the State Manager page of the Systems Manager console.

Warning

An AWS Identity and Access Management (IAM) user, group, or role with permission to create an association that targets a resource group of Amazon EC2 instances automatically has root-level control of all instances in the group. Only trusted administrators should be permitted to create associations.


Choose all instances
Use this option to target all instances in the current AWS account and Region. When you run the request, the system locates and attempts to create the association on all instances in the current AWS account and Region. When the system initially creates the association, it runs the association. After this initial run, the system runs the association according to the schedule you specified. If you create new instances, the system automatically applies the association, runs it immediately, and then runs it according to the schedule.

Rate controls

You can control the execution of an association on your instances by specifying a concurrency value and an error threshold. The concurrency value specifies how many instances are allowed to run the association simultaneously. An error threshold specifies how many association executions can fail before Systems Manager sends a command to each instance configured with that association to stop running the association. The command stops the association from running until the next scheduled execution. The concurrency and error threshold features are collectively called rate controls.

Concurrency

Concurrency helps to limit the impact on your instances by allowing you to specify that only a certain number of instances can process an association at one time. You can specify either an absolute number of instances, for example 20, or a percentage of the target set of instances, for example 10%.

State Manager concurrency has the following restrictions and limitations:

- If you choose to create an association by using targets, but you don't specify a concurrency value, then State Manager automatically enforces a maximum concurrency of 50 instances.
- If new instances that match the target criteria come online while an association that uses concurrency is running, then the new instances run the association if the concurrency value is not exceeded. If the concurrency value is exceeded, then the instances are ignored during the current association execution interval. The instances run the association during the next scheduled interval while conforming to the concurrency requirements.
- If you update an association that uses concurrency, and one or more instances are processing that association when it is updated, then any instance that is running the association is allowed to complete. Those associations that haven't started are stopped. After running associations complete, all target instances immediately run the association again because it was updated. When the association runs again, the concurrency value is enforced.

Error thresholds

An error threshold specifies how many association executions are allowed to fail before Systems Manager sends a command to each instance configured with that association. The command stops the association
from running until the next scheduled execution. You can specify either an absolute number of errors, for example 10, or a percentage of the target set, for example 10%.

If you specify an absolute number of three errors, for example, State Manager sends the stop command when the fourth error is returned. If you specify 0, then State Manager sends the stop command after the first error result is returned.

If you specify an error threshold of 10% for 50 associations, then State Manager sends the stop command when the sixth error is returned. Associations that are already running when an error threshold is reached are allowed to complete, but some of these associations might fail. To ensure that there aren't more errors than the number specified for the error threshold, set the Concurrency value to 1 so that associations proceed one at a time.

State Manager error thresholds have the following restrictions and limitations:

- Error thresholds are enforced for the current interval.
- Information about each error, including step-level details, is recorded in the association history.
- If you choose to create an association by using targets, but you don't specify an error threshold, then State Manager automatically enforces a threshold of 100% failures.

Create an association

The following procedures describe how to create a State Manager association by using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for Windows PowerShell.

**Important**

The following procedures describe how to create an association that uses either a Command or a Policy document. For information about creating an association that uses an Automation document, see Running Automation workflows with triggers using State Manager (p. 319).

When you create a State Manager association, by default, the system immediately runs it on the specified instances or targets. After the initial run, the association runs in intervals according to the schedule that you defined and according to the following rules:

- Associations only run on instances that are online when the interval starts. Offline instances are skipped.
- State Manager attempts to run the association on all specified or targeted instances during an interval.
- If an association doesn't run during an interval (because, for example, a concurrency value limited the number of instances that could process the association at one time), then State Manager attempts to run the association during the next interval.
- State Manager records history for all skipped intervals. You can view the history on the Execution History tab.

**Note**

If you don't want an association to run immediately after you create it, you can choose the **Apply association only at the next specified Cron interval** option in the Systems Manager console.

The following procedure describes how to use targets and rate controls when creating an association. For more information about these features, see About targets and rate controls in State Manager associations (p. 867).

**Warning**

An AWS Identity and Access Management (IAM) user, group, or role with permission to create an association that targets a resource group of Amazon EC2 instances automatically has root-level control of all instances in the group. Only trusted administrators should be permitted to create associations.
Create an association (console)

The following procedure describes how to use the Systems Manager console to create a State Manager association.

To create a State Manager association

2. In the navigation pane, choose State Manager, and then choose Create association.
3. In the Name field, specify a name. This is optional, but recommended. A name helps you remember the purpose of the association. For example, you could specify Automatically_update_AWSPVDrivers_on_us-west-2_instances for an association with that purpose. Spaces aren't allowed in the name.
4. In the Document list, choose the option next to a document name. Note the document type. This procedure applies only to Command and Policy documents. For information about creating an association that uses an Automation document, see Running Automation workflows with triggers using State Manager (p. 319).
5. For Parameters, specify the required input parameters.
6. For Targets, choose an option. For information about using targets, see About targets and rate controls in State Manager associations (p. 867).
7. In the Specify schedule section, choose either On Schedule or No schedule. If you choose On Schedule, use the buttons provided to create a cron or rate schedule for the association.
   
   If you don't want the association to run immediately after you create it, choose Apply association only at the next specified Cron interval.
8. In the Advanced options section use the Compliance severity to choose a severity level for the association. Compliance reporting indicates whether the association state is compliant or noncompliant, along with the severity level you indicate here. For more information, see About State Manager association compliance (p. 700).
9. In the Rate control section, choose options to control how the association runs on multiple instances. For more information about using rate controls, see About targets and rate controls in State Manager associations (p. 867).

   In the Concurrency section, choose an option:

   • Choose targets to enter an absolute number of targets that can run the association simultaneously.
   • Choose percentage to enter a percentage of the target set that can run the association simultaneously.

   In the Error threshold section, choose an option:

   • Choose errors to enter an absolute number of errors that are allowed before State Manager stops running associations on additional targets.
   • Choose percentage to enter a percentage of errors that are allowed before State Manager stops running associations on additional targets.
10. (Optional) For Output options, to save the command output to a file, select the Enable writing output to S3 box. Type the bucket and prefix (folder) names in the boxes.

   Note
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.
11. Choose **Create Association**.

**Note**
If you delete the association you created, the association no longer runs on any targets of that association.

**Create an association (command line)**

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to create a State Manager association. This section includes several examples that show how to use targets and rate controls. Targets and rate controls enable you to assign an association to dozens or hundreds of instances while controlling the execution of those associations. For more information about targets and rate controls, see *About targets and rate controls in State Manager associations* (p. 867).

**Before you begin**

The `targets` parameter is an array of search criteria that targets instances using a `Key,Value` combination that you specify. If you plan to create an association on dozens or hundreds of instance by using the `targets` parameter, review the following targeting options before you begin the procedure.

**Target a few instances by specifying IDs**

```bash
--targets Key=InstanceIds,Values=instance-id-1,instance-id-2,instance-id-3
```

```bash
--targets Key=InstanceIds,Values=i-02573cafEXAMPLE,i-0471e04240EXAMPLE,i-07782c72faEXAMPLE
```

**Target instances by using Amazon EC2 tags**

```bash
--targets Key=tag:tag-key,Values=tag-value-1,tag-value-2,tag-value-3
```

```bash
--targets Key=tag:Environment,Values=Development,Test,Pre-production
```

**Target instances by using AWS Resource Groups**

```bash
--targets Key=resource-groups:Name,Values=resource-group-name
```

```bash
--targets Key=resource-groups:Name,Values=WindowsInstancesGroup
```

**Target all instances in the current AWS account and Region**

```bash
--targets Key=InstanceIds,Values=* 
```

**Note**

When you create an association, you specify when the schedule runs. You must specify the schedule by using a cron or rate expression. For more information about cron and rate expressions, see *Cron and rate expressions for associations* (p. 1193).

**To create an association**

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.
For information, see Install or upgrade AWS command line tools (p. 57).

2. Use the following format to create a command that creates a State Manager association.

**Linux**

```bash
aws ssm create-association \
   --targets target_options \
   --name document_name \
   --schedule "cron_or_rate_expression" \
   --parameters (if any) \
   --max-concurrency a_number_of_instances_or_a_percentage_of_target_set \
   --max-errors a_number_of_errors_or_a_percentage_of_target_set
```

**Windows**

```bash
aws ssm create-association ^ \
   --targets target_options ^ \
   --name document_name ^ \
   --schedule "cron_or_rate_expression" ^ \
   --parameters (if any) ^ \
   --max-concurrency a_number_of_instances_or_a_percentage_of_target_set ^ \
   --max-errors a_number_of_errors_or_a_percentage_of_target_set
```

**PowerShell**

```powershell
New-SSMAssociation \
   -AssociationName document_name \
   -Target target_options \
   -ScheduleExpression "cron_or_rate_expression" \
   -Parameters (if any) \
   -MaxConcurrency a_number_of_instances_or_a_percentage_of_target_set \
   -MaxError a_number_of_errors_or_a_percentage_of_target_set
```

The following example creates an association on instances tagged with "Environment,Linux". The association uses the AWS-UpdateSSMAgent document to update SSM Agent on the targeted instances at 2:00 every Sunday morning. This association runs simultaneously on 10 instances maximum at any given time. Also, this association stops running on more instances for a particular execution interval if the error count exceeds 5. For compliance reporting, this association is assigned a severity level of Medium.

**Linux**

```bash
aws ssm create-association \
   --association-name Update_SSM_Agent_Linux \
   --targets Key=tag:Environment,Values=Linux \
   --name AWS-UpdateSSMAgent \
   --compliance-severity "MEDIUM" \
   --schedule "cron(0 2 ? * SUN *)" \
   --max-errors "5" \
   --max-concurrency "10"
```

**Windows**

```bash
aws ssm create-association ^ \
   --association-name Update_SSM_Agent_Linux ^ \
   --targets Key=tag:Environment,Values=Linux ^ \
   --name AWS-UpdateSSMAgent ^
```
Working with associations

--compliance-severity "MEDIUM" ^
--schedule "cron(0 2 ? * SUN *)" ^
--max-errors "5" ^
--max-concurrency "10"

PowerShell

New-SSMAssociation  
-AssociationName Update_SSM_Agent_Linux  
-Name AWS-UpdateSSMAgent  
-Target @{
    "Key"="tag:Environment"
    "Values"="Linux"
}  
-ScheduleExpression "cron(0 2 ? * SUN *)"  
-MaxConcurrency 10  
-MaxError 5  
-ComplianceSeverity MEDIUM

The following example targets instance IDs by specifying a wildcard value (*). This enables Systems Manager to create an association on all instances in the current account and AWS Region. This association runs simultaneously on 10 instances maximum at any given time. Also, this association stops running on more instances for a particular execution interval if the error count exceeds 5. For compliance reporting, this association is assigned a severity level of Medium. This association runs only at the specified Cron schedule. It doesn't run immediately after the association is created.

Linux

aws ssm create-association 
  --association-name Update_SSM_Agent_Linux 
  --name "AWS-UpdateSSMAgent" 
  --targets "Key=instanceids,Values="*"" 
  --compliance-severity "MEDIUM" 
  --schedule "cron(0 2 ? * SUN *)" 
  --max-errors "5" 
  --max-concurrency "10" 
  --apply-only-at-cron-interval

Windows

aws ssm create-association  
  --association-name Update_SSM_Agent_Linux  
  --name "AWS-UpdateSSMAgent"  
  --targets "Key=instanceids,Values="*""  
  --compliance-severity "MEDIUM"  
  --schedule "cron(0 2 ? * SUN *)"  
  --max-errors "5"  
  --max-concurrency "10"  
  --apply-only-at-cron-interval

PowerShell

New-SSMAssociation  
-AssociationName Update_SSM_Agent_All  
-Name AWS-UpdateSSMAgent  
-Target @{
    "Key"="InstanceIds"
    "Values"="*"
}  

874
The following example creates an association on instances in AWS Resource Groups. The group is named "HR-Department". The association uses the AWS-UpdateSSMAgent document to update SSM Agent on the targeted instances at 2:00 every Sunday morning. This association runs simultaneously on 10 instances maximum at any given time. Also, this association stops running on more instances for a particular execution interval if the error count exceeds 5. For compliance reporting, this association is assigned a severity level of Medium. This association runs only at the specified Cron schedule. It doesn't run immediately after the association is created.

**Linux**

```bash
aws ssm create-association
    --association-name Update_SSM_Agent_Linux
    --targets Key=resource-groups:Name,Values=HR-Department
    --name AWS-UpdateSSMAgent
    --compliance-severity "MEDIUM"
    --schedule "cron(0 2 ? * SUN *)"
    --max-errors "5"
    --max-concurrency "10"
    --apply-only-at-cron-interval
```

**Windows**

```bash
aws ssm create-association
    --association-name Update_SSM_Agent_Linux
    --targets Key=resource-groups:Name,Values=HR-Department
    --name AWS-UpdateSSMAgent
    --compliance-severity "MEDIUM"
    --schedule "cron(0 2 ? * SUN *)"
    --max-errors "5"
    --max-concurrency "10"
    --apply-only-at-cron-interval
```

**PowerShell**

```powershell
New-SSMAssociation
    -AssociationName Update_SSM_Agent_Linux
    -Name AWS-UpdateSSMAgent
    -Target @{
        "Key"="resource-groups:Name"
        "Values"="HR-Department"
    }
    -ScheduleExpression "cron(0 2 ? * SUN *)"
    -MaxConcurrency 10
    -MaxError 5
    -ComplianceSeverity MEDIUM
    -ApplyOnlyAtCronInterval
```

**Note**

If you delete the association you created, the association no longer runs on any targets of that association. Also, if you specified the `apply-only-at-cron-interval` parameter, you can reset this option. To do so, specify the `no-apply-only-at-cron-interval` parameter when
you update the association from the command line. This parameter forces the association to run immediately after updating the association and according to the interval specified.

**Edit and create a new version of an association**

You can edit an association to specify a new name, schedule, severity level, or targets. You can also choose to write the output of the command to an Amazon S3 bucket. After you edit an association, Systems Manager creates a new version. You can view different versions after editing, as described in the following procedures.

The following procedures describe how to edit and create a new version of an association using the AWS Systems Manager console, AWS Command Line Interface (AWS CLI), and AWS Tools for PowerShell.

**Edit an association (console)**

The following procedure describes how to use the Systems Manager console to edit and create a new version of an association.

**Note**

This procedure requires that you have write access to an existing S3 bucket. If you have not used Amazon S3 before, be aware that you will incur charges for using Amazon S3. For information about how to create a bucket, see Create a Bucket.

**To edit a State Manager association**

2. In the navigation pane, choose State Manager.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose State Manager.
3. Choose the association you created in the previous procedure, and then choose Edit.
4. In the Name field, type a new name. For example, type TestHostnameAssociation2.
5. In the Specify schedule section, choose a new option. For example, choose CRON schedule builder, and then choose Every 1 hour.
6. (Optional) For Output options, to save the command output to a file, select the Enable writing output to S3 box. Type the bucket and prefix (folder) names in the boxes.
   **Note**
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.
7. Choose Edit association.
8. In the Associations page, choose the name of the association you just edited, and then choose the Versions tab. The system lists each version of the association you created and edited.
10. Choose the name of the S3 bucket you specified for storing command output, and then choose the folder named with the ID of the instance that ran the association. (If you chose to store output in a folder in the bucket, open it first.)
11. Drill down several levels, through the awsrunPowerShell folder, to the stdout file.
12. Choose Open or Download to view the host name.
Edit an association (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to edit and create a new version of an association.

To edit a State Manager association

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Use the following format to create a command to edit and create a new version of an existing State Manager association.

   Linux

   ```bash
   aws ssm update-association
      --association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE
      --association-name association_name
      --parameters (if any)
      --output-location S3Location='{OutputS3Region=region,OutputS3BucketName=bucketname,OutputS3KeyPrefix=keyprefix}'
      --scheduleexpression "cron_or_rate_expression"
   ```

   **Important**
   To retain existing parameter values of your association, such as association name or compliance severity, you must specify these values when you update the association. If you don't specify these parameter values when you update an association, the new association version uses no values. For example, if your existing association has a cron schedule but you don't specify --schedule-expression when updating, the new association version will not have a schedule expression.

   Windows

   ```bash
   aws ssm update-association ^
      --association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE ^
      --association-name association_name ^
      --parameters (if any) ^
      --output-location S3Location='{OutputS3Region=region,OutputS3BucketName=bucketname,OutputS3KeyPrefix=keyprefix}' ^
      --scheduleexpression "cron_or_rate_expression"
   ```

   **Important**
   To retain existing parameter values of your association, such as association name or compliance severity, you must specify these values when you update the association. If you don't specify these parameters when you update an association, the new association version uses the default values (none). For example, if your existing association has a cron schedule but you don't specify --schedule-expression when updating, the new association version will not have a schedule expression.

   PowerShell

   ```powershell
   Update-SSMAssociation `^`  
      -AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE `^`  
      -AssociationName document_name `^`  
      -Parameter (if any) `^`  
      -S3Location_OutputS3BucketName bucket_name `^`  
      -S3Location_OutputS3KeyPrefix key_prefix `^`  
      -S3Location_OutputS3Region region `^`
   ```
Important

To retain existing parameter values of your association, such as association name or compliance severity, you must specify these values when you update the association. If you don’t specify these parameters when you update an association, the new association version uses no values. For example, if your existing association has a cron schedule but you don’t specify `--ScheduleExpression` when updating, the new association version will not have a schedule expression.

The following example updates an existing association to change the name to `TestHostnameAssociation2`. The new association version runs every hour and writes the output of commands to the specified Amazon S3 bucket.

Linux

```bash
aws ssm update-association  
  --association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE  
  --association-name TestHostnameAssociation2  
  --parameters commands="echo Association"  
  --output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=statemanager,OutputS3KeyPrefix=logs}'  
  --schedule-expression "cron(0 */1 * * ? *)"
```

Windows

```bash
aws ssm update-association ^  
  --association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE ^  
  --association-name TestHostnameAssociation2 ^  
  --parameters commands="echo Association" ^  
  --output-location S3Location='{OutputS3Region=us-east-1,OutputS3BucketName=statemanager,OutputS3KeyPrefix=logs}' ^  
  --schedule-expression "cron(0 */1 * * ? *)"
```

PowerShell

```powershell
Update-SSMAssociation `  
  -AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE `  
  -AssociationName TestHostnameAssociation2 `  
  -Parameter @{"commands"="echo Association"} `  
  -S3Location_OutputS3BucketName statemanager `  
  -S3Location_OutputS3KeyPrefix logs `  
  -S3Location_OutputS3Region us-east-1 `  
  -ScheduleExpression "cron(0 */1 * * ? *)"
```

3. To view the new version of the association, run the following command.

Linux

```bash
aws ssm describe-association  
  --association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE
```

Windows

```bash
aws ssm describe-association ^  
  --association-id b85ccafe-9f02-4812-9b81-01234EXAMPLE
```
PowerShell

```powershell
Get-SSMAssociation
   -AssociationId b85ccafe-9f02-4812-9b81-01234EXAMPLE | Select-Object *
```

The system returns information like the following.

Linux

```json
{
   "AssociationDescription": {
      "ScheduleExpression": "cron(0 */1 * * *)",
      "OutputLocation": {
         "S3Location": {
            "OutputS3KeyPrefix": "logs",
            "OutputS3BucketName": "statemanager",
            "OutputS3Region": "us-east-1"
         }
      },
      "Name": "AWS-RunPowerShellScript",
      "Parameters": {
         "commands": [
            "echo Association"
         ],
      },
      "LastExecutionDate": 1559316400.338,
      "Overview": {
         "Status": "Success",
         "DetailedStatus": "Success",
         "AssociationStatusAggregatedCount": {}
      },
      "AssociationId": "b85ccafe-9f02-4812-9b81-01234EXAMPLE",
      "DocumentVersion": "$DEFAULT",
      "LastSuccessfulExecutionDate": 1559316400.338,
      "LastUpdateAssociationDate": 1559316389.753,
      "Date": 1559314038.532,
      "AssociationVersion": "2",
      "AssociationName": "TestHostnameAssociation2",
      "Targets": [
         {
            "Values": [
               "Windows"
            ],
            "Key": "tag:Environment"
         }
      ]
   }
}
```

Windows

```json
{
   "AssociationDescription": {
      "ScheduleExpression": "cron(0 */1 * * *)",
      "OutputLocation": {
         "S3Location": {
            "OutputS3KeyPrefix": "logs",
            "OutputS3BucketName": "statemanager",
            "OutputS3Region": "us-east-1"
         }
      }
   }
}
```
### Viewing association histories

You can view all executions for a specific association ID by using the `DescribeAssociationExecutions` API action. This action allows you to quickly see the status, detailed status, results, last execution time, and more information for a State Manager association. This API action also includes filters to help you...
quickly locate associations according to the criteria you specify. For example, you can specify an exact date and time, and use a GREATER_THAN filter to view only those executions that were processed after the specified date and time.

If, for example, an association execution failed, you can drill down into the details of a specific execution by using the DescribeAssociationExecutionTargets API action. This action shows you the resources, such as instance IDs, where the association ran and the various association statuses. You can then quickly see which resource or instance failed to run an association. With the resource ID you can then view the command execution details to see exactly which step in a command failed.

The examples in this section also include information about how to use the StartAssociationsOnce API action to run an association immediately and only one time. You can use this API action when you investigate failed association executions. If you see that an association failed, you can make a change on the resource, and then immediately run the association to see if the change on the resource allows the association to run successfully.

Viewing association histories (console)

Use the following procedure to view the execution history for a specific association ID and then view execution details for one or more resources.

To view execution history for a specific association ID

2. Choose State Manager.
3. In the Association id field, choose an association for which you want to view the history.
4. Choose the View details button.
5. Choose the Execution history tab.
6. Choose an association for which you want to view resource-level execution details. For example, choose an association that shows a status of Failed. You can then view the execution details for the instances that failed to run the association.

   **Note**
   Use the search box filters to locate the execution for which you want to view details.

7. Choose an execution ID. The Association execution targets page opens. This page shows all of the resources that ran the association.
8. Choose a resource ID to view specific information about that resource.

   **Note**
   Use the search box filters to locate the resource for which you want to view details.

9. If you are investigating an association that failed to run, you can use the Apply association now button to run an association immediately and only one time. After you made changes on the resource where the association failed to run, choose the Association ID link in the navigation breadcrumb.
10. Choose the Apply association now button. After the execution is complete, verify that the association execution succeeded.

Viewing association histories (command line)

The following procedure describes how to use the AWS CLI (on Linux or Windows) or AWS Tools for PowerShell to view the execution history for a specific association ID. Following this, the procedure describes how to view execution details for one or more resources.
To view execution history for a specific association ID

1. Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to view a list of executions for a specific association ID.

   Linux

   ```bash
   aws ssm describe-association-executions \
   --association-id ID \
   --filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN
   ```

   **Note**
   
   This command includes a filter to limit the results to only those executions that occurred after a specific date and time. If you want to view all executions for a specific association ID, remove the `--filters` parameter and `Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN` value.

   Windows

   ```bash
   aws ssm describe-association-executions ^
   --association-id ID ^
   --filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN
   ```

   **Note**
   
   This command includes a filter to limit the results to only those executions that occurred after a specific date and time. If you want to view all executions for a specific association ID, remove the `--filters` parameter and `Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=GREATER_THAN` value.

   PowerShell

   ```powershell
   Get-SSMAssociationExecution `\n   -AssociationId ID `\n   -Filter @{"Key="CreatedTime";"Value="2019-06-01T19:15:38.372Z";"Type="GREATER_THAN"}
   ```

   **Note**
   
   This command includes a filter to limit the results to only those executions that occurred after a specific date and time. If you want to view all executions for a specific association ID, remove the `-Filter` parameter and `@{"Key="CreatedTime";"Value="2019-06-01T19:15:38.372Z";"Type="GREATER_THAN"}` value.

   The system returns information like the following.

   Linux

   ```json
   {  
     "AssociationExecutions":[
       {  
         "Status":"Success",
         "DetailedStatus":"Success",
         "AssociationId":"c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
         "ExecutionId":"76a5a04f-caf6-490c-b448-92c02EXAMPLE",
       
   ```

   ```json
     }
   ```

   ```json
     }]
   ```

882
Windows

```json
{
  "AssociationExecutions": [
    {
      "Status": "Success",
      "DetailedStatus": "Success",
      "AssociationId": "c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
      "ExecutionId": "f91b72e0-50da-4021-8b35-f95dfEXAMPLE",
      "CreatedTime": 1523986028.219,
      "AssociationVersion": "1"
    },
    {
      "Status": "Success",
      "DetailedStatus": "Success",
      "AssociationId": "c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
      "ExecutionId": "791b72e0-f0da-4021-8b35-f95dfEXAMPLE",
      "CreatedTime": 1523984226.074,
      "AssociationVersion": "1"
    },
    {
      "Status": "Success",
      "DetailedStatus": "Success",
      "AssociationId": "c336d2ab-09de-44ba-8f6a-6136cEXAMPLE",
      "ExecutionId": "ecec60fa-6bb0-4d26-98c7-140308EXAMPLE",
      "CreatedTime": 1523982404.013,
      "AssociationVersion": "1"
    }
  ]
}
```

PowerShell

```
AssociationId          : c336d2ab-09de-44ba-8f6a-6136cEXAMPLE
AssociationVersion     : 1
CreatedAt              : 8/18/2019 2:00:50 AM
DetailedStatus        : Success
ExecutionId            : 76a5a04f-caf6-490c-b448-92c02EXAMPLE
LastExecutionDate      : 1/1/0001 12:00:00 AM
ResourceCountByStatus : {Success=1}
Status                 : Success
```
You can limit the results by using one or more filters. The following example returns all associations that were run before a specific date and time.

Linux

```bash
cat
aws ssm describe-association-executions \
  --association-id ID \
  --filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=LESS_THAN
```

Windows

```bash
cat
aws ssm describe-association-executions ^
  --association-id ID ^
  --filters Key=CreatedTime,Value="2018-04-10T19:15:38.372Z",Type=LESS_THAN
```

PowerShell

```bash
cat
Get-SSMAssociationExecution `  
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE `  
  -Filter @("Key"="CreatedTime","Value"="2019-06-01T19:15:38.372Z","Type"="LESS_THAN")
```

The following returns all associations that were successfully run after a specific date and time.

Linux

```bash
cat
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
```

Windows

```bash
cat
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
```
PowerShell

```
Get-SSMAssociationExecution
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE
  -Filter @{
    "Key"="CreatedTime";
    "Value"="2019-06-01T19:15:38.372Z";
    "Type"="GREATER_THAN"
  },
  @{
    "Key"="Status";
    "Value"="Success";
    "Type"="EQUAL"
  }
```

3. Run the following command to view all targets where the specific execution ran.

**Linux**

```
aws ssm describe-association-execution-targets
  --association-id ID
  --execution-id ID
```

**Windows**

```
aws ssm describe-association-execution-targets
  --association-id ID
  --execution-id ID
```

**PowerShell**

```
Get-SSMAssociationExecutionTarget
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE
  -ExecutionId 76a5a04f-caf6-490c-b448-92c02EXAMPLE
```

You can limit the results by using one or more filters. The following example returns information about all targets where the specific association failed to run.

**Linux**

```
aws ssm describe-association-execution-targets
  --association-id ID
  --execution-id ID
  --filters Key=Status,Value="Failed"
```

**Windows**

```
aws ssm describe-association-execution-targets
  --association-id ID
  --execution-id ID
  --filters Key=Status,Value="Failed"
```

**PowerShell**

```
Get-SSMAssociationExecutionTarget
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE
```

885
The following example returns information about a specific managed instance where an association failed to run.

**Linux**

```bash
aws ssm describe-association-execution-targets \
  --association-id ID \
  --execution-id ID \
  --filters Key=Status,Value=Failed Key=ResourceId,Value="i-02573cafcfEXAMPLE" Key=ResourceType,Value=ManagedInstance
```

**Windows**

```bash
aws ssm describe-association-execution-targets ^
  --association-id ID ^
  --execution-id ID ^
  --filters Key=Status,Value=Failed Key=ResourceId,Value="i-02573cafcfEXAMPLE" Key=ResourceType,Value=ManagedInstance
```

**PowerShell**

```bash
Get-SSMAssociationExecutionTarget \
  -AssociationId 14bea65d-5ccc-462d-a2f3-e99c8EXAMPLE \
  -ExecutionId 76a5a04f-caf6-490c-b448-92c02EXAMPLE \
  -Filter @{
    "Key"="Status";
    "Value"="Success"
  },
  @{
    "Key"="ResourceId";
    "Value"="i-02573cafcfEXAMPLE"
  },
  @{
    "Key"="ResourceType";
    "Value"="ManagedInstance"
  }
```

4. If you are investigating an association that failed to run, you can use the `StartAssociationsOnce` API action to run an association immediately and only one time. After you change the resource where the association failed to run, run the following command to run the association immediately and only one time.

**Linux**

```bash
aws ssm start-associations-once \
  --association-id ID
```

**Windows**

```bash
aws ssm start-associations-once ^
  --association-id ID
```
Creating associations that run MOF files

You can run Managed Object Format (MOF) files to enforce a desired state on Windows Server managed instances with State Manager by using the AWS-ApplyDSCMofs SSM document. The AWS-ApplyDSCMofs document has two execution modes. With the first mode, you can configure the association to scan and report if the managed instances are currently in the desired state defined in the specified MOF files. In the second mode, you can run the MOF files and change the configuration of your instances based on the resources and their values defined in the MOF files. The AWS-ApplyDSCMofs document enables you to download and run MOF configuration files from Amazon Simple Storage Service (Amazon S3), a local share, or from a secure web site with an HTTPS domain.

State Manager logs and reports the status of each MOF file execution during each association run. State Manager also reports the output of each MOF file execution as a compliance event which you can view on the AWS Systems Manager Compliance page.

MOF file execution is built on Windows PowerShell Desired State Configuration (PowerShell DSC). PowerShell DSC is a declarative platform used for configuration, deployment, and management of Windows systems. PowerShell DSC allows administrators to describe, in simple text documents called DSC configurations, how they want a server to be configured. A PowerShell DSC configuration is a specialized PowerShell script that states what to do, but not how to do it. Running the configuration produces a MOF file. The MOF file can be applied to one or more servers to achieve the desired configuration for those servers. PowerShell DSC resources do the actual work of enforcing configuration. For more information, see Windows PowerShell Desired State Configuration Overview.

Topics

- Using Amazon S3 to store artifacts (p. 888)
- Resolving credentials in MOF files (p. 888)
- Using tokens in MOF files (p. 889)
- Prerequisites (p. 890)
- Creating an association that runs MOF files (p. 890)
- Troubleshooting (p. 893)
- Viewing DSC resource compliance details (p. 895)
Using Amazon S3 to store artifacts

If you are using Amazon S3 to store PowerShell modules, MOF files, compliance reports, or status reports, then the IAM role used by SSM Agent must have GetObject and ListBucket permissions on the bucket. If you don’t provide these permissions, the system returns an Access Denied error. Also note the following important information about storing artifacts in Amazon S3.

- If the bucket is in a different AWS account, then you must create a bucket resource policy that grants the account (or the IAM role) GetObject and ListBucket permissions.
- If you want to use custom DSC resources, you can download these resources from an Amazon S3 bucket. You can also install them automatically from the PowerShell gallery.
- If you are using Amazon S3 as a module source, then you need to upload the module as a Zip file in the following case-sensitive format: ModuleName_ModuleVersion.zip. For example: MyModule_1.0.0.zip.
- All files must be in the bucket root. Folder structures are not supported.

Resolving credentials in MOF files

Credentials are resolved by using AWS Secrets Manager or AWS Systems Manager Parameter Store (p. 198). This allows you to set up automatic credential rotation. This also enables DSC to automatically propagate credentials to your servers without redeploying MOFs.

To use a Secrets Manager secret in a configuration, create a PSCredential object where the Username is the SecretId or SecretARN of the secret containing the credential. You can specify any value for the password. The value is ignored. Here is an example:

```powershell
Configuration MyConfig
{
    $ss = ConvertTo-SecureString -String 'a_string' -AsPlaintext -Force
    $credential = New-Object PSCredential('a_secret_or_ARN', $ss)

    Node localhost
    {
        File $file_name
        {
            DestinationPath = 'C:\MyFile.txt'
            SourcePath = '\FileServer\Share\MyFile.txt'
            Credential = $credential
        }
    }
}
```

You must then compile your MOF using the PsAllowPlaintextPassword setting in configuration data. This is OK because the credential only contains a label.

In Secrets Manager, ensure that the instance has GetSecretValue access in an IAM Managed Policy, and optionally in the Secret Resource Policy if one exists. In order to work with DSC, the secret must be in the following format:

```json
{
    'Username': 'a_name',
    'Password': 'a_password'
}
```

The secret can have other properties (for example, properties used for rotation), but it must at least have the username and password properties.

We recommended that you use a multi-user rotation method, where you have two different usernames and passwords, and the rotation AWS Lambda function flips between them. This method allows you to have multiple active accounts while eliminating the risk of locking out a user during rotation.
Using tokens in MOF files

Tokens give you the ability to modify resource property values after the MOF has been compiled. This enables you to reuse common MOF files on multiple servers that require very similar configurations.

Token substitution only works for Resource Properties of type String. However, if your resource has a nested CIM instance property, it will also resolve tokens from String properties in that CIM instance. You can’t use token substitution for numerals or arrays.

For example, consider a scenario where you’re using the xComputerManagement resource and you want to rename the computer using DSC. Normally you would need a dedicated MOF file for that machine. However, with token support, you can create a single MOF file and apply it to all of your instances. In the ComputerName property, instead of hard coding the computer name into the MOF, you can use an Instance Tag type token. The value is resolved during MOF parsing. For example:

```powershell
Configuration MyConfig
{
    xComputer Computer
    {
        ComputerName = '{tag:ComputerName}'
    }
}
```

You then set a tag on either the managed instance in the AWS Systems Manager console, or an Amazon EC2 tag in the EC2 console. When you run the document, the script substitutes the `{tag:ComputerName}` token for the value of the instance tag.

You can also combine multiple tags into a single property, for example:

```powershell
Configuration MyConfig
{
    File MyFile
    {
        DestinationPath = '{env:TMP}\{tag:ComputerName}'
        Type = 'Directory'
    }
}
```

There are 5 different types of tokens you can use:

- **tag**: Amazon EC2 or managed instance tags
- **tagb64**: This is the same as tag, but the system use base64 to decode the value. This allows you to use special characters in tag values.
- **env**: Resolves Environment variables.
- **ssm**: Systems Manager Parameter Store values. Only String and Secure String types are supported.
- **tagssm**: This is the same as tag, but if the tag is not set on the instance, the system tries to resolve the value from an SSM Parameter with the same name. This is useful in situations when you want a 'default global value' but you want to be able to override it on a single instance (for example, one-box deployments).

Here is Parameter Store example that uses the `ssm` token type.

```powershell
File MyFile
{
    DestinationPath = "C:\ProgramData\ConnectionData.txt"
    Content = "{ssm:%servicePath%/ConnectionData}"
}
```
Tokens play an important role in reducing redundant code by making MOF files generic and reusable. If you can avoid server-specific MOF file, then there's no need for a MOF building service. A MOF building service increases costs, slows provisioning time, and increases the risk of configuration drift between grouped instances due to differing module versions being installed on the build server when their MOFs were compiled.

**Prerequisites**

Before you create an association that runs MOF files, verify that your managed instances have the following prerequisites installed:

- Windows PowerShell version 5.0 or later. For more information, see [Windows PowerShell System Requirements](https://docs.microsoft.com/en-us/powershell/scripting/serversystem-requirements) on Microsoft.com.
- [AWS Tools for Windows PowerShell](https://docs.aws.amazon.com/ssm/latest/ug/install-macos.html) version 3.3.261.0 or later.
- SSM Agent version 2.2 or later.

**Creating an association that runs MOF files**

**To create an association that runs MOF files**

2. In the navigation pane, choose **State Manager**, and then choose **Create association**.
3. In the **Name** field, specify a name. This is optional, but recommended. A name can help you understand the purpose of the association when you created it. Spaces aren't allowed in the name.
4. In the **Document** list, choose **AWS-ApplyDSCMofs**.
5. In the **Parameters** section, specify your choices for the required and optional input parameters.

   a. **Mofs To Apply**: Specify one or more MOF files to run when this association runs. Use commas to separate a list of MOF files. You can specify the following options for locating MOF file.

      - An Amazon S3 bucket name. Bucket names must use lowercase letters. Specify this information by using the following format:

        ```
        s3:bucket_name:MOF_file_name.mof
        ```

        If you want to specify an AWS Region, then use the following format:

        ```
        s3:bucket_Region:bucket_name:MOF_file_name.mof
        ```

      - A secure web site. Specify this information by using the following format:

        ```
        https://domain_name/MOF_file_name.mof
        ```

        Here is an example:

        ```
        https://AWS.Amazon.com/TestMOF.mof
        ```

      - A file system on a local share. Specify this information by using the following format:

        ```
        \server_name\shared_folder_name\MOF_file_name.mof
        ```

        Here is an example:
b. **Service Path:** (Optional) A service path is either an Amazon S3 bucket prefix where you want to write reports and status information. Or, a service path is a path for Parameter Store parameter-based tags. When resolving parameter-based tags, the system uses `{ssm:%servicePath %/parameter_name}` to inject the servicePath value into the parameter name. For example, if your service path is "WebServers/Production" then the system resolves the parameter as: WebServers/Production/parameter_name. This is useful for when you are running multiple environments in the same account.

c. **Report Bucket Name:** (Optional) Enter the name of an Amazon S3 bucket where you want to write compliance data. Reports are saved in this bucket in JSON format.

   **Note**
   You can prefix the bucket name with a Region where the bucket is located. Here's an example: us-west-2:MyMOFBucket. If you are using a proxy for Amazon S3 endpoints in a specific region that does not include us-east-1, then you must prefix the bucket name with a region. If the bucket name is not prefixed, it will automatically discover the bucket region using the us-east-1 endpoint.

d. **Mof Operation Mode:** Choose State Manager behavior when running the `AWS-ApplyDSCMofs` association:

   - **Apply:** Correct instance configurations that aren't compliant.
   - **ReportOnly:** Don't correct instance configurations, but instead log all compliance data and report instances that aren't compliant.

e. **Status Bucket Name:** (Optional) Enter the name of an Amazon S3 bucket where you want to write MOF execution status information. These status reports are singleton summaries of the most recent compliance run of an instance. This means that the report is overwritten the next time the association runs MOF files.

   **Note**
   You can prefix the bucket name with a Region where the bucket is located. Here's an example: us-west-2:MyMOFBucket. If you are using a proxy for Amazon S3 endpoints in a specific region that does not include us-east-1, then you must prefix the bucket name with a region. If the bucket name is not prefixed, it will automatically discover the bucket region using the us-east-1 endpoint.

f. **Module Source Bucket Name:** (Optional) Enter the name of an Amazon S3 bucket that contains PowerShell module files. If you specify **None**, then you must choose **True** for the next option, **Allow PS Gallery Module Source**.

   **Note**
   You can prefix the bucket name with a Region where the bucket is located. Here's an example: us-west-2:MyMOFBucket. If you are using a proxy for Amazon S3 endpoints in a specific region that does not include us-east-1, then you must prefix the bucket name with a region. If the bucket name is not prefixed, it will automatically discover the bucket region using the us-east-1 endpoint.

g. **Allow PS Gallery Module Source:** (Optional) Choose **True** to download PowerShell modules from [https://www.powershellgallery.com/](https://www.powershellgallery.com/). If you choose **False**, then you must specify a source for the previous option, **ModuleSourceBucketName**.

h. **Proxy Uri:** (Optional) Use this option to download MOF files from a proxy server.

i. **Reboot Behavior:** (Optional) Specify one of the following reboot behaviors if your MOF file execution requires rebooting:

   - **AfterMof:** Reboots the instance after all MOF executions are complete. Even if multiple MOF executions request reboots, the system waits until all MOF executions are complete to reboot.
• **Immediately**: Reboots the instance whenever a MOF execution requests it. If running multiple MOF files that request reboots, then the instance will be rebooted multiple times.

• **Never**: Instances are not rebooted, even if the MOF execution explicitly requests a reboot.

j. **Use Computer Name For Reporting**: (Optional) Enable this option to use the name of the computer when reporting compliance information. The default value is **false**, which means that the system uses the instance ID when reporting compliance information.

k. **EnableVerbose Logging**: (Optional) We recommend that you enable verbose logging when deploying MOF files for the first time.

   **Important**
   When enabled, verbose logging writes more data to your Amazon S3 bucket than standard association execution logging. This can result in slower performance and possibly higher storage charges for Amazon S3. To mitigate storage size issues, we recommend that you enable lifecycle policies on your Amazon S3 bucket. For more information, see How Do I Create a Lifecycle Policy for an S3 Bucket? in the Amazon Simple Storage Service Console User Guide.

l. **Enable Debug Logging**: (Optional) We recommend that you enable debug logging if you need to troubleshoot MOF failures. We also recommend that you disable this option for normal use.

   **Important**
   When enabled, debug logging writes more data to your Amazon S3 bucket than standard association execution logging. This can result in slower performance and possibly higher storage charges for Amazon S3. To mitigate storage size issues, we recommend that you enable lifecycle policies on your Amazon S3 bucket. For more information, see How Do I Create a Lifecycle Policy for an S3 Bucket? in the Amazon Simple Storage Service Console User Guide.

m. **Compliance Type**: (Optional) Specify the compliance type to use when reporting compliance information. The default compliance type is **Custom:DSC**. If you create multiple associations that run MOF files, then be sure to specify a different compliance type for each association. If you don't, each additional association that uses **Custom:DSC** will overwrite the existing compliance data.

n. **Pre Reboot Script**: (Optional) Specify a script to run if the configuration has indicated that a reboot is necessary. The script runs before the reboot. The script must be a single line. If you need to add additional lines, separate lines by using semicolons.

6. In the **Targets** section, choose either **Specifying tags** or **Manually Selecting Instance**. If you choose to target resources by using tags, then enter a tag key and a tag value in the fields provided. For more information about using targets, see About targets and rate controls in State Manager associations (p. 867).

7. In the **Specify schedule** section, choose either **On Schedule** or **No schedule**. If you choose **On Schedule**, then use the buttons provided to create a cron or rate schedule for the association.

8. In the **Advanced options** section:

   • In **Compliance severity**, choose a severity level for the association. Compliance reporting will indicate whether the association state is compliant or non-compliant, along with the severity level you indicate here. For more information, see About State Manager association compliance (p. 700).

9. In the **Rate control** section, configure options for running State Manager associations across of fleet of managed instances. For more information about these options, see About targets and rate controls in State Manager associations (p. 867).

   In the **Concurrency** section, choose an option:

   • Choose **targets** to enter an absolute number of targets that can run the association simultaneously.
• Choose **percentage** to enter a percentage of the target set that can run the association simultaneously.

In the **Error threshold** section, choose an option:

• Choose **errors** to enter an absolute number of errors allowed before State Manager stops running associations on additional targets.
• Choose **percentage** to enter a percentage of errors allowed before State Manager stops running associations on additional targets.

10. **(Optional)** For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Type the bucket and prefix (folder) names in the boxes.

**Note**
The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

11. Choose **Create Association**.

State Manager creates and immediately runs the association on the specified instances or targets. After the initial execution, the association runs in intervals according to the schedule that you defined and according to the following rules:

• Associations are only run on instances that are online when the interval starts. Offline instances are skipped.
• State Manager attempts to run the association on all configured instances during an interval.
• If an association is not run during an interval (because, for example, a concurrency value limited the number of instances that could process the association at one time), then State Manager attempts to run the association during the next interval.
• State Manager records history for all skipped intervals. You can view the history on the **Execution History** tab.

**Note**
The AWS-ApplyDSCMofs is a Systems Manager command document. This means that you can also run this document by using Run Command. For more information, see Running commands using Systems Manager Run Command (p. 833).

**Troubleshooting**

This section includes information to help you troubleshoot issues creating associations that run MOF files.

**Enable enhanced logging**

As a first step to troubleshooting, enable enhanced logging. More specifically, do the following:

• Verify that the association is configured to write command output to either Amazon S3 or Amazon CloudWatch Logs.
• Set the **Enable Verbose Logging** parameter to True.
• Set the **Enable Debug Logging** parameter to True.
With verbose and debug logging enabled, the **Stdout** output file includes details about the script execution. This output file can help you identify where the script failed. The **Stderr** output file contains errors that occurred during the script execution.

**Common problems**

This section includes information about common problems that can occur when creating associations that run MOF files and steps to troubleshoot these issues.

**My MOF was not applied**

If State Manager failed to apply the association to your instances, then start by reviewing the **Stderr** output file. This file can help you understand the root cause of the issue. Also, verify the following:

- The instance has the required access permissions to all MOF-related Amazon S3 buckets. Specifically:
  - **s3:GetObject permissions**: This is required for MOF files in private Amazon S3 buckets as well as custom modules in Amazon S3 buckets.
  - **s3:PutObject permission**: This is required to write compliance reports and compliance status to Amazon S3 buckets.
- If you are using tags, then ensure that the instance has the required IAM policy. Using tags requires the instance IAM role to have a policy allowing the `ec2:DescribeInstances` and `ssm:ListTagsForResource` actions.
- Ensure that the instance has the expected tags or SSM parameters assigned.
- Ensure that the tags or SSM parameters aren't misspelled.
- Try applying the MOF locally on the instance to make sure there isn't an issue with the MOF file itself.

**My MOF seemed to fail, but the Systems Manager execution was successful**

If the AWS-ApplyDSCMofs document successfully ran, then the Systems Manager execution status shows **Success**. This status does not reflect the compliance status of your instance against the configuration requirements in the MOF file. To view the compliance status of your instances, view the compliance reports. You can view a JSON report in the Amazon S3 Report Bucket. This applies to Run Command and State Manager executions. Also, for State Manager, you can view compliance details on the Systems Manager Compliance page.

**Stderr states: Name resolution failure attempting to reach service**

This error indicates that the script can't reach a remote service. Most likely, the script can't reach Amazon S3. This issue most often occurs when the script attempts to write compliance reports or compliance status to the Amazon S3 bucket supplied in the document parameters. Typically, this error occurs when a computing environment uses a firewall or transparent proxy that includes a whitelist. To resolve this issue:

- Use region-specific bucket syntax for all Amazon S3 bucket parameters. For example, the **Mofs to Apply** parameter should be formatted as follows:
  
  ```
  s3:bucket-region:bucket-name:mof-file-name.mof.
  ```
  
  Here is an example: `s3:us-west-2:my-bucket:my-mof.mof`
  
  The Report, Status, and Module Source bucket names should be formatted as follows:
  
  ```
  bucket-region:bucket-name.
  ```
  
  Here is an example: `us-west-1:my-bucket`
  
  - If Region-specific syntax does not fix the problem, then make sure that the targeted instance(s) can access Amazon S3 in the desired region. To verify this:
    1. Find the endpoint name for Amazon S3 in the appropriate Amazon S3 Region. For information, see Amazon S3 Service Endpoints in the Amazon Web Services General Reference.
2. Log on to the target instance and run the following ping command:

```
ping s3.s3-region.amazonaws.com
```

If the ping failed, it means that either Amazon S3 is down, or a firewall/transparent proxy is blocking access to the Amazon S3 region, or the instance can't access the internet.

**Viewing DSC resource compliance details**

Systems Manager captures compliance information about DSC resource failures in the Amazon Simple Storage Service (Amazon S3) **Status Bucket** you specified when you ran the AWS-ApplyDSCMofs document. Searching for information about DSC resource failures in an Amazon S3 bucket can be time consuming. Instead, you can quickly view this information in the Systems Manager **Compliance** page.

The **Compliance resources summary** section displays a count of resources that failed. In the following example, the **ComplianceType** is **Custom:DSC** and one resource is non-compliant.

**Note**

Custom:DSC is the default **ComplianceType** value in the AWS-ApplyDSCMofs document. This value is customizable.

The **Details overview for resources** section displays information about the AWS resource with the non-compliant DSC resource. This section also includes the MOF name, script execution steps, and (when applicable) a **View output** link to view detailed status information.

The **View output** link displays the last 4,000 characters of the detailed status. Systems Manager starts with the exception as the first element, and then scans back through the verbose messages...
and prepends as many as it can until it reaches the 4000 character limit. This process displays the log messages that were output prior to the exception being thrown, which are the most relevant messages for troubleshooting.

For information about how to view compliance information, see AWS Systems Manager Configuration Compliance (p. 696).

Situations that affect compliance reporting

If the State Manager association fails, then no compliance data is reported. More specifically, if a MOF fails to process, then Systems Manager doesn’t report any compliance items because the associations fail. For example, if Systems Manager attempts to download a MOF from an Amazon S3 bucket that the instance doesn’t have permission to access, then the association fails and no compliance data is reported.

If a resource in a second MOF fails, then Systems Manager does report compliance data. For example, if a MOF tries to create a file on a drive that doesn’t exist, then Systems Manager reports compliance because the AWS-ApplyDSCMofs document is able to process completely, which means the association successfully runs.

Creating associations that run Ansible playbooks

You can create State Manager associations that run Ansible playbooks by using the AWS-ApplyAnsiblePlaybooks document. This document offers the following benefits for running playbooks:

- Support for running complex playbooks
- Support for downloading playbooks from GitHub and Amazon Simple Storage Service (Amazon S3)
- Support for compressed playbook structure
- Enhanced logging
• Ability to specify which playbook to run when playbooks are bundled

Note
Systems Manager includes two SSM documents that enable you to create State Manager associations that run Ansible playbooks: AWS-RunAnsiblePlaybook and AWS-ApplyAnsiblePlaybooks. The AWS-RunAnsiblePlaybook document is deprecated. It remains available in Systems Manager for legacy purposes. We recommend that you use the AWS-ApplyAnsiblePlaybooks document because of the enhancements described here.

Support for running complex playbooks
The AWS-ApplyAnsiblePlaybooks document supports bundled, complex playbooks because it copies the entire file structure to a local directory before executing the specified main playbook. You can provide source playbooks in Zip files or in a directory structure. The Zip file or directory can be stored in GitHub or Amazon S3.

Support for downloading playbooks from GitHub
The AWS-ApplyAnsiblePlaybooks document uses the aws:downloadContent plugin to download playbook files. Files can be stored in GitHub in a single file or as a combined set of playbook files. To download content from GitHub, you must specify information about your GitHub repository in JSON format. Here is an example:

```
{
    "owner":"TestUser",
    "repository":"GitHubTest",
    "path":"scripts/python/test-script",
    "getOptions":"branch:master",
    "tokenInfo":"{"ssm-secure:secure-string-token}"
}
```

Support for downloading playbooks from Amazon S3
You can also store and download Ansible playbooks in Amazon S3 as either a single .zip file or a directory structure. To download content from Amazon S3, you must specify the path to the file. Here are two examples:

Example 1: Download a specific playbook file

```
{
    "path":"https://s3.amazonaws.com/aws-execute-ansible-test/ansible/playbook.yml"
}
```

Example 2: Download the contents of a directory

```
{
    "path":"https://s3.amazonaws.com/aws-execute-ansible-test/ansible/webservers/"
}
```

Important
If you specify Amazon S3, then the AWS Identity and Access Management (IAM) instance profile on your managed instances must be configured with the AmazonS3ReadOnlyAccess policy. For more information, see Create an IAM instance profile for Systems Manager (p. 30).

Support for compressed playbook structure
The AWS-ApplyAnsiblePlaybooks document enables you to run compressed .zip files in the downloaded bundle. The document checks if the downloaded files contain a compressed file in .zip
format. If a .zip is found, the document automatically decompresses the file and then runs the specified Ansible automation.

**Enhanced logging**

The AWS-ApplyAnsiblePlaybooks document includes an optional parameter for specifying different levels of logging. Specify `-v` for low verbosity, `-vv` or `-vvv` for medium verbosity, and `-vvvv` for debug level logging. These options directly map to Ansible verbosity options.

**Ability to specify which playbook to run when playbooks are bundled**

The AWS-ApplyAnsiblePlaybooks document includes a required parameter for specifying which playbook to run when multiple playbooks are bundled. This option provides flexibility for running playbooks to support different use cases.

**Installed dependencies**

If you specify `True` for the `InstallDependencies` parameter, then Systems Manager verifies that the following dependencies are installed on your instances. If one or more of these dependencies are not found, then Systems Manager automatically installs them.

- **Ubuntu/Debian**: APT-get (Package Management), Python 3, Ansible, Unzip
- **Amazon Linux**: Ansible
- **RHEL**: Python 3, Ansible, Unzip

**Create an association that runs Ansible playbooks (console)**

The following procedure describes how to use the Systems Manager console to create a State Manager association that runs Ansible playbooks by using the AWS-ApplyAnsiblePlaybooks document.

**To create an association that runs Ansible playbooks (console)**

2. In the navigation pane, choose **State Manager**, and then choose **Create association**.
3. For **Name**, specify a name that helps you remember the purpose of the association.
4. In the **Document** list, choose **AWS-ApplyAnsiblePlaybooks**.
5. In the **Parameters** section, for **Source Type**, choose either **GitHub** or **S3**.
   
   **GitHub**
   
   If you choose **GitHub**, enter repository information in the following format:

   ```json
   {
   "owner": "user_name",
   "repository": "name",
   "path": "path_to_directory_or_playbook_to_download",
   "getOptions": "branch:branch_name",
   "tokenInfo": "{{(Optional)_token_information}}"
   }
   ```

   **S3**
   
   If you choose **S3**, enter path information in the following format:

   ```json
   {
   "path": "https://s3.amazonaws.com/path_to_directory_or_playbook_to_download"
   ```
6. For **Install Dependencies**, choose an option.

7. (Optional) For **Playbook File**, enter a file name. If the playbook is contained in a Zip file, then you must specify a relative path to the Zip file.

8. (Optional) For **Extra Variables**, enter variables that you want State Manager to send to Ansible at runtime.

9. (Optional) For **Check**, choose an option.

10. (Optional) For **Verbose**, choose an option.

11. For **Targets**, choose an option. For information about using targets, see About targets and rate controls in State Manager associations (p. 867).

12. In the **Specify schedule** section, choose either **On schedule** or **No schedule**. If you choose **On schedule**, then use the buttons provided to create a cron or rate schedule for the association.

13. In the **Advanced options** section, for **Compliance severity**, choose a severity level for the association. Compliance reporting indicates whether the association state is compliant or noncompliant, along with the severity level you indicate here. For more information, see About State Manager association compliance (p. 700).

14. In the **Rate control** section, configure options to run State Manager associations across a fleet of managed instances. For information about using rate controls, see About targets and rate controls in State Manager associations (p. 867).

   In the **Concurrency** section, choose an option:

   - Choose **targets** to enter an absolute number of targets that can run the association simultaneously.
   - Choose **percentage** to enter a percentage of the target set that can run the association simultaneously.

   In the **Error threshold** section, choose an option:

   - Choose **errors** to enter an absolute number of errors that are allowed before State Manager stops running associations on additional targets.
   - Choose **percentage** to enter a percentage of errors that are allowed before State Manager stops running associations on additional targets.

15. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Type the bucket and prefix (folder) names in the boxes.

   **Note**
   
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

16. Choose **Create Association**.

   **Note**
   
   If you use tags to create an association on one or more target instances, and then you remove the tags from an instance, that instance no longer runs the association. The instance is disassociated from the State Manager document.

**Create an association that runs Ansible playbooks (CLI)**

The following procedure describes how to use the AWS CLI to create a State Manager association that runs Ansible playbooks by using the **AWS-ApplyAnsiblePlaybooks** document.
To create an association that runs Ansible playbooks (CLI)

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run one of the following commands to create an association that runs Ansible playbooks by targeting instances using Amazon EC2 tags. Command (A) specifies GitHub as the source type. Command (B) specifies Amazon S3 as the source type.

(A) GitHub source

**Linux**

```bash
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" \ 
--targets Key=tag:OS,Values=Linux \ 
--parameters '{"SourceType":"GitHub","SourceInfo":
 ["\"owner\":\"owner_name\", \"repository\": \"name\", 
 \"getOptions\": \"branch:master\"],"InstallDependencies":
 ["True_or_False"],"PlaybookFile":\"file_name.yml","ExtraVariables":
 ["key/value_pairs_separated_by_a_space"],"Check":\"True_or_False","Verbose":\"-v,-vv,-vvv, or -vvvv\"} \ 
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

**Windows**

```bash
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" ^ 
--targets Key=tag:OS,Values=Linux ^ 
--parameters '{"SourceType":"GitHub","SourceInfo":
 ["\"owner\":\"owner_name\", \"repository\": \"name\", 
 \"getOptions\": \"branch:master\"],"InstallDependencies":
 ["True_or_False"],"PlaybookFile":\"file_name.yml","ExtraVariables":
 ["key/value_pairs_separated_by_a_space"],"Check":\"True_or_False","Verbose":\"-v,-vv,-vvv, or -vvvv\"} ^ 
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

Here is an example:

```bash
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" \ 
--targets Key=tag:OS,Values=Linux \ 
--parameters '{"SourceType":"GitHub","SourceInfo":
 ["\"owner\":\"ansibleDocumentTest\", \"repository\": \"Ansible\", 
 \"getOptions\": \"branch:master\"],"InstallDependencies":
 ["True"],"PlaybookFile":\"hello-world-playbook.yml","ExtraVariables":["SSM=True","Check":\"False",\"Verbose":\"-v\"],"Verbose":\"-v\"} \ 
--association-name "AnsibleAssociation" --schedule-expression "cron(0 2 ? * SUN *)"
```

(B) S3 source

**Linux**

```bash
aws ssm create-association --name "AWS-ApplyAnsiblePlaybooks" \ 
--targets Key=tag:OS,Values=Linux \ 
--parameters '{"SourceType":"S3","SourceInfo":
 ["\"path\":\"https://s3.amazonaws.com/path_to_zip_file_directory_or_playbook_to_download\"]},"InstallDependencies":
 ["True_or_False"],"PlaybookFile":\"file_name.yml","ExtraVariables":
 ["key/value_pairs_separated_by_a_space"],"Check":\"True_or_False","Verbose":\"-v,-vv,-vvv, or -vvvv\"} \ 
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

900
Creating associations that run Chef recipes

You can create State Manager associations that run Chef recipes by using the AWS-ApplyChefRecipes document. You can target Linux-based Systems Manager managed nodes with the AWS-ApplyChefRecipes document. This document offers the following benefits for running Chef recipes:

- Supports multiple releases of Chef (Chef 11 through Chef 14).
- Automatically installs the Chef client software on target instances.
- Optionally runs Systems Manager compliance checks (p. 696) on target instances, and stores the results of compliance checks in an S3 bucket.
- Runs multiple cookbooks and recipes in a single run of the document.
- Optionally runs recipes in why-run mode, to show which recipes will change on target instances without making changes.
- Optionally applies custom JSON attributes to chef-client runs.

You can use GitHub or S3 buckets as sources for Chef cookbooks and recipes that you specify in an AWS-ApplyChefRecipes document.

Prerequisites: Set up your association, repository, and cookbooks

Before you create an AWS-ApplyChefRecipes document, prepare your Chef cookbooks and cookbook repository. If you don't already have a Chef cookbook that you want to use, you can get started by using...
a test HelloWorld cookbook that AWS has prepared for you. The AWS-ApplyChefRecipes document already points to this cookbook by default. Your cookbooks should be set up similarly to the following directory structure. In the following example, jenkins and nginx are examples of Chef cookbooks that are available in the Chef Supermarket on the Chef website.

Though AWS cannot officially support cookbooks on the Chef Supermarket website, many of them work with the AWS-ApplyChefRecipes document. The following are examples of criteria to check when you are testing a community cookbook:

- The cookbook should support the Linux-based operating systems of the Systems Manager managed nodes that you are targeting.
- The cookbook should be valid for the Chef client version (Chef 11 through Chef 14) that you use.
- The cookbook is compatible with Chef Infra Client, and, does not require a Chef server.

Verify that you can reach the Chef.io website, so that any cookbooks you specify in your run list can be installed when the Systems Manager document runs. Using a nested cookbooks folder is supported, but not required; you can store cookbooks directly under the root level.

### cookbooks (optional level)
   ### jenkins
      #   ### metadata.rb
      #   ### recipes
   ### nginx
      ### metadata.rb
      ### recipes

**Important**

Before you create a State Manager association that runs Chef recipes, be aware that the document run installs the Chef client software on your Systems Manager managed nodes, unless you set the value of Chef client version to None. This action uses an installation script from Chef to install Chef components on your behalf. Before you run an AWS-ApplyChefRecipes document, be sure your enterprise can comply with any applicable legal requirements, including license terms applicable to the use of Chef software. For more information, see the Chef website.

Systems Manager can deliver compliance reports to an S3 bucket, the Systems Manager console, or make compliance results available in response to Systems Manager API commands. To run Systems Manager compliance reports, the instance profile attached to Systems Manager managed instances must have permissions to write to the S3 bucket. The instance profile must have permissions to use the Systems Manager PutComplianceItem API. For more information about Systems Manager compliance, see AWS Systems Manager Configuration Compliance (p. 696).

**Logging the document run**

When you run a Systems Manager document by using a State Manager association, you can configure the association to choose the output of the document run, and you can send the output to Amazon S3 or Amazon CloudWatch Logs. To help ease troubleshooting when an association has finished running, verify that the association is configured to write command output to either an S3 bucket or CloudWatch Logs. For more information, see Create an association (p. 870).

**Use GitHub as a cookbook source**

The AWS-ApplyChefRecipes document uses the aws:downloadContent (p. 1065) plugin to download cookbooks. To download content from GitHub, specify information about your GitHub repository to the document in JSON format. The following is an example:

```json
{
```
Use Amazon S3 as a cookbook source

You can also store and download Chef cookbooks in Amazon S3 as either a single .zip or tar.gz file or a directory structure. To download content from Amazon S3, you must specify the path to the file. Here are two examples:

Example 1: Download a specific cookbook

```
{
  "path":"https://s3.amazonaws.com/chef-cookbooks/HelloWorld.zip"
}
```

Example 2: Download the contents of a directory

```
{
  "path":"https://s3.amazonaws.com/chef-cookbooks-test/HelloWorld"
}
```

Important

If you specify Amazon S3, the AWS Identity and Access Management (IAM) instance profile on your managed instances must be configured with the AmazonS3ReadOnlyAccess policy. For more information, see Create an IAM instance profile for Systems Manager (p. 30).

Topics

- Create an association that runs Chef recipes (console) (p. 903)
- Create an association that runs Chef recipes (CLI) (p. 905)
- Viewing Chef resource compliance details (p. 907)

Create an association that runs Chef recipes (console)

The following procedure describes how to use the Systems Manager console to create a State Manager association that runs Chef cookbooks by using the AWS-ApplyChefRecipes document.

2. In the navigation pane, choose State Manager, and then choose Create association.
3. For Name, enter a name that helps you remember the purpose of the association.
4. In the Document list, choose AWS-ApplyChefRecipes.
5. In Parameters, for Source Type, choose either GitHub or S3.
6. In Source info, enter cookbook source information in one of the following formats.
   - If you chose GitHub in step 5, enter repository information in the following format:

     ```
     {
       "owner":"user_name",
       "repository":"name",
       "path":"path_to_directory_or_cookbook_to_download",
       "getOptions":"branch:branch_name",
       "tokenInfo":"{{Optional_token_information}}"
     }
     ```

     If you also enter repository information, make sure you also enter the branch name.
     ```
     {
       "owner":"TestUser",
       "repository":"GitHubCookbookRepository",
       "path":"cookbooks/HelloWorld",
       "getOptions":"branch:master",
       "tokenInfo":"{{ssm-secure:secure-string-token}}"
     }
     ```

     If you enter the path only, then the branch name is set to the default master branch.

8. If you chose S3 in step 5, enter path information in the following format:

     ```
     {
       "path":"https://s3.amazonaws.com/chef-cookbooks/HelloWorld.zip"
     }
     ```

9. If you also enter repository information, then you must also enter the branch name.

9. Review the information on the Create association page, and then choose Create association.
b. If you chose S3 in step 5, enter path information in the following format:

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

7. In Run list, list the recipes that you want to run in the following format, separating each recipe with a comma as shown. Do not include a space after the comma.

```plaintext
recipe[cookbook_name1::recipe_name],recipe[cookbook_name2::recipe_name]
```

8. (Optional) In JSON attributes content, add any custom JSON that contains attributes you want the Chef client to pass to your target instances.

The **JSON attributes content** parameter is best used for the following purposes:

- When you want to override only a small number of attributes, and you do not otherwise need to use custom cookbooks.
  
  Custom JSON can help you avoid the extra work of setting up and maintaining a cookbook repository to override only a few attributes.

- Values that are expected to vary.

  For example, if your Chef cookbooks configure a third-party application that accepts payments, you can use custom JSON to specify the payment endpoint URL. If the third-party software manufacturer changes the payment endpoint URL, you can use custom JSON to update the payment endpoint to the new URL.

9. For **Chef client version**, specify a Chef version. Valid values are 11, 12, 13, 14, or None. If you specify 11 through 14, Systems Manager installs the correct Chef client version on your target instances. If you specify None, Systems Manager does not install the Chef client on target instances before running the document's recipes. The default value is 14.

10. (Optional) For **Chef client arguments**, specify additional arguments that are supported for the version of Chef you are using. To learn more about supported arguments, run `chef-client -h` on an instance that is running the Chef client.

11. (Optional) Enable **Why-run** to show changes that will be made to target instances if the recipes are run, without actually changing target instances.

12. For **Compliance severity**, choose the severity of Systems Manager Configuration Compliance results that you want reported. Compliance reporting indicates whether the association state is compliant or noncompliant, along with the severity level you specify. Configuration Compliance reports are stored in an S3 bucket that you specify as the value of the **Compliance report bucket** parameter (step 15). For more information about Configuration Compliance, see Working with Configuration Compliance (p. 699) in this guide.

   Compliance scans measure drift between configuration that is specified in your Chef recipes and instance resources. Valid values are Critical, High, Medium, Low, Informational, Unspecified, or None. To skip compliance reporting, choose None.

13. For **Compliance type**, specify the compliance type for which you want results reported. Valid values are Association for State Manager associations, or Custom:custom_type. The default value is Custom:Chef.

14. For **Compliance report bucket**, enter the name of an S3 bucket in which to store information about every Chef run performed by this document, including resource configuration and Configuration Compliance results.

15. In **Rate control**, configure options to run State Manager associations across a fleet of managed instances. For information about using rate controls, see About targets and rate controls in State Manager associations (p. 867).
In **Concurrency**, choose an option:

- Choose **targets** to enter an absolute number of targets that can run the association simultaneously.
- Choose **percentage** to enter a percentage of the target set that can run the association simultaneously.

In **Error threshold**, choose an option:

- Choose **errors** to enter an absolute number of errors that are allowed before State Manager stops running associations on additional targets.
- Choose **percentage** to enter a percentage of errors that are allowed before State Manager stops running associations on additional targets.

16. (Optional) For **Output options**, to save the command output to a file, select the **Enable writing output to S3** box. Type the bucket and prefix (folder) names in the boxes.

   **Note**
   The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

17. Choose **Create Association**.

### Create an association that runs Chef recipes (CLI)

The following procedure describes how to use the AWS CLI to create a State Manager association that runs Chef cookbooks by using the AWS-ApplyChefRecipes document.

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run one of the following commands to create an association that runs Chef cookbooks by targeting instances using Amazon EC2 tags. Command (A) uses GitHub as the source type. Command (B) uses Amazon S3 as the source type.

   **(A) GitHub source**

   **Linux**

   ```bash
   aws ssm create-association --name "AWS-ApplyChefRecipes" \
   --targets Key=tag:TagKey,Values=TagValue \
   --parameters '{"SourceType": "GitHub","SourceInfo": {"owner \
   ": "owner_name", 
   "repository": "name", 
   "path": 
   "path_to_directory_or_cookbook_to_download"}, 
   "getOptions": 
   {"branch:branch_name"}, 
   "RunList": [{"recipe[cookbook_name1::recipe_name]"}, 
   "recipe[cookbook_name2::recipe_name]"}], 
   "JsonAttributesContent": [{"Custom_JSON"}], 
   "ChefClientVersion": "version_number", 
   "ChefClientArguments": [{"<chef_client_arguments>"]}, 
   "WhyRun": true_or_false, 
   "ComplianceSeverity": 
   ["severity_value"], 
   "ComplianceType": ["Custom:Chef"], 
   "ComplianceReportBucket": ["S3_bucket_name"]}' \
   --association-name "name" --schedule-expression "cron_or_rate_expression"
   ```

   **Windows**

   ```bash
   aws ssm create-association --name "AWS-ApplyChefRecipes" ^
   ```

---

905
Here is an example:

```bash
aws ssm create-association --name "AWS-ApplyChefRecipes" |
--targets Key=tag:OS,Values=Linux
--parameters '{"SourceType": ["GitHub"], "SourceInfo": "{\"owner\": "ChefRecipeTest", "repository": "ChefCookbooks", "path": "cookbooks/HelloWorld"}, "getOptions": {"branch": "master"}, "RunList": [{"recipe[HelloWorld::HelloWorldRecipe]"}, {"recipe[HelloWorld::InstallApp]"}], "JsonAttributesContent": "{\"colors\": {\"foreground\": \"light-blue\", \"background\": \"dark-gray\"}}", "ChefClientVersion": ["14"], "ChefClientArguments": [], "WhyRun": false, "ComplianceSeverity": ["Medium"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket": ["ChefComplianceResultsBucket"]}' |
--association-name "MyChefAssociation" --schedule-expression "cron(0 2 ? * SUN *)"
```

(B) S3 source

**Linux**

```bash
aws ssm create-association --name "AWS-ApplyChefRecipes" |
--targets Key=tag:OS,Values=Linux
--parameters '{"SourceType": ["S3"], "SourceInfo": "{\"path\": "https://s3.amazonaws.com/path_to_Zip_file_directory_or_cookbook_to_download\"}"}, "RunList": [{"recipe[cookbook_name1::recipe_name]"}, "recipe[cookbook_name2::recipe_name]"], "JsonAttributesContent": "{\"state\": \"visible\"}, "ChefClientVersion": ["14"], "ChefClientArguments": [], "WhyRun": true_or_false, "ComplianceSeverity": ["severity_value"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket": ["S3_bucket_name"]}' |
--association-name "name" --schedule-expression "cron_or_rate_expression"
```

**Windows**

```bash
aws ssm create-association --name "AWS-ApplyChefRecipes" |
--targets Key=tag:OS,Values=Windows
--parameters '{"SourceType": ["S3"], "SourceInfo": "{\"path\": "https://s3.amazonaws.com/path_to_Zip_file_directory_or_cookbook_to_download\"}"}, "RunList": [{"recipe[cookbook_name1::recipe_name]"}, "recipe[cookbook_name2::recipe_name]"], "JsonAttributesContent": "{\"state\": \"visible\"}, "ChefClientVersion": ["14"], "ChefClientArguments": [], "WhyRun": true_or_false, "ComplianceSeverity": ["severity_value"], "ComplianceType": ["Custom:Chef"], "ComplianceReportBucket": ["S3_bucket_name"]}' |
--association-name "name" --schedule-expression "cron_or_rate_expression"
```
aws ssm create-association --name "AWS-ApplyChefRecipes" ^
--targets "Key=tag:OS,Values= Windows" ^
--parameters '{"SourceType": ["S3"], "SourceInfo": ["{\"path\": \"https://s3.amazonaws.com/ ChefCookbooks/HelloWorld\"}"], "RunList": ["{\"recipe[HelloWorld::HelloWorldRecipe]\", 
{\"recipe[HelloWorld::InstallApp]\"}"}, "JsonAttributesContent": ["{\"state\": 
"visible\"}, "colors": {"foreground": "light-blue","background": "dark-gray \"}}], "ChefClientVersion": ["14"], "ChefClientArguments": ["{--fips}"], "WhyRun": false, "ComplianceSeverity": ["Medium"], "ComplianceType": ["Custom:Chef"],
"ComplianceReportBucket": ["ChefComplianceResultsBucket"]}' ^
--association-name "name" --schedule-expression "cron(0 2 ? * SUN *)"

**Note**

State Manager associations do not support all cron and rate expressions. For more information about creating cron and rate expressions for associations, see Reference: Cron and rate expressions for Systems Manager (p. 1189).

The system attempts to create the association on the instances and immediately apply the state.

3. Run the following command to view an updated status of the association you just created.

    ```bash
    aws ssm describe-association --association-id "ID"
    ```

**Viewing Chef resource compliance details**

Systems Manager captures compliance information about Chef-managed resources in the Amazon Simple Storage Service (Amazon S3) **Compliance report bucket** value that you specified when you ran the AWS-ApplyChefRecipes document. Searching for information about Chef resource failures in an S3 bucket can be time consuming. Instead, you can view this information on the Systems Manager Compliance page.

A Systems Manager Compliance scan collects information about resources on your managed nodes that were created or checked in the most recent Chef run. The resources can include files, directories, systemctl services, yum packages, templated files, gem packages, and dependent cookbooks, among others.

The **Compliance resources summary** section displays a count of resources that failed. In the following example, the **ComplianceType** is **Custom:Chef** and one resource is noncompliant.

**Note**

Custom:Chef is the default **ComplianceType** value in the AWS-ApplyChefRecipes document. This value is customizable.
The **Details overview for resources** section shows information about the AWS resource that is not in compliance. This section also includes the Chef resource type against which compliance was run, severity of issue, compliance status, and links to more information when applicable.

### Details overview for resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>ID</th>
<th>Resource type</th>
<th>Compliance type</th>
<th>Overall severity</th>
<th>Overall status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>i-0</td>
<td>ManagedInstance</td>
<td>Custom:Chef</td>
<td>Critical</td>
<td>Compliant</td>
</tr>
</tbody>
</table>

#### Compliance rule

<table>
<thead>
<tr>
<th>ID</th>
<th>Compliance type</th>
<th>Resource ID</th>
<th>Severity</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws-site:install-nginx:nginx</td>
<td>Custom:Chef</td>
<td>i-0</td>
<td>Critical</td>
<td>Compliant</td>
</tr>
<tr>
<td>aws-site:install-nginx:nginx</td>
<td>Custom:Chef</td>
<td>i-0</td>
<td>Critical</td>
<td>Compliant</td>
</tr>
<tr>
<td>aws-site:install-nginx:/var/www/html/</td>
<td>Custom:Chef</td>
<td>i-0</td>
<td>Critical</td>
<td>Compliant</td>
</tr>
<tr>
<td>aws-site:install-nginx:/etc/nginx/nginx.conf</td>
<td>Custom:Chef</td>
<td>i-0</td>
<td>Critical</td>
<td>Compliant</td>
</tr>
<tr>
<td>aws-site:deploy-app:/usr/share/nginx/html/index.html</td>
<td>Custom:Chef</td>
<td>i-0</td>
<td>Critical</td>
<td>Compliant</td>
</tr>
</tbody>
</table>

**View output** shows the last 4,000 characters of the detailed status. Systems Manager starts with the exception as the first element, and then finds verbose messages and shows as many as it can until it reaches the 4000 character limit. This process displays the log messages that were output before the exception was thrown, which are the most relevant messages for troubleshooting.

For information about how to view compliance information, see AWS Systems Manager Configuration Compliance (p. 696).

**Association failures affect compliance reporting**

If the State Manager association fails, no compliance data is reported. For example, if Systems Manager attempts to download a Chef cookbook from an S3 bucket that the instance doesn’t have permission to access, the association fails, and Systems Manager reports no compliance data.

### Automatically update SSM Agent (CLI)

The following procedure walks you through the process of creating a State Manager association using the AWS Command Line Interface (AWS CLI). The association automatically updates the SSM Agent
according to a schedule that you specify. For more information about the SSM Agent, see Working with SSM Agent (p. 63).

**Note**
To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

**Before you begin**

Before you complete the following procedure, verify that you have at least one running EC2 instance for Linux or Windows Server that is configured for Systems Manager. For more information, see Systems Manager prerequisites (p. 14).

**Note**
If you create an association by using either the AWS CLI or AWS Tools for Windows PowerShell, use the `--Targets` parameter to target instances, as shown in the following example. Don't use the `--InstanceId` parameter. The `--InstanceId` parameter is a legacy parameter.

**To create an association for automatically updating SSM Agent**

1. Install and configure the AWS CLI, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

2. Run the following command to create an association by targeting instances using Amazon EC2 tags. The `Schedule` parameter sets a schedule to run the association every Sunday morning at 2:00 a.m. (UTC).

   ```bash
   aws ssm create-association --targets Key=tag:TagKey,Values=TagValue --name AWS-UpdateSSMAgent --schedule-expression "cron(0 2 ? * SUN *)"
   
   **Note**
   State Manager associations do not support all cron and rate expressions. For more information about creating cron and rate expressions for associations, see Reference: Cron and rate expressions for Systems Manager (p. 1189).
   
   If you want, you can also target multiple instances by specifying instances IDs in a comma-separated list.

   ```bash
   aws ssm create-association --targets Key=instanceids,Values=InstanceId,InstanceId,InstanceId --name your document name --schedule-expression "cron(0 2 ? * SUN *)"
   ```

   The system returns information like the following.

```
{
    "AssociationDescription": {
        "ScheduleExpression": "cron(0 2 ? * SUN *)",
        "Name": "AWS-UpdateSSMAgent",
        "Overview": {
            "Status": "Pending",
            "DetailedStatus": "Creating"
        },
        "AssociationId": "123.\ldots\ldots.\ldots",
        "DocumentVersion": "#DEFAULT",
        "LastUpdateAssociationDate": 1504034257.98,
        "Date": 1504034257.98,
        "AssociationVersion": "1",
        "Targets": [
            {
                "Values": ["InstanceId",
```
The system attempts to create the association on the instance(s) and immediately apply the state. The association status shows Pending.

3. Run the following command to view an updated status of the association you just created.

```bash
aws ssm list-associations
```

**Note**

If your instances aren't running the most recent version of the SSM Agent, the status shows Failed. This is expected behavior. When a new version of SSM Agent is published, the association automatically installs the new agent, and the status shows Success.

---

### Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server (console)

Amazon Windows AMIs contain a set of drivers to permit access to virtualized hardware. These drivers are used by Amazon EC2 to map instance store and Amazon EBS volumes to their devices. We recommend that you install the latest drivers to improve stability and performance of your EC2 instances for Windows Server. For more information about PV drivers, see AWS PV Drivers.

The following walkthrough shows you how to configure a State Manager association to automatically download and install new AWS PV drivers when the drivers become available.

**Before you begin**

Before you complete the following procedure, verify that you have at least one EC2 instance for Windows Server running that is configured for Systems Manager. For more information, see Systems Manager prerequisites (p. 14).

**To create a State Manager association that automatically updates PV drivers**

1. Open the Amazon EC2 console, expand Systems Manager Services in the navigation pane, and then choose State Manager.
2. Choose Create Association.
3. In the **Association Name** field, type a descriptive name.
4. In the **Select Document** list, choose AWS-ConfigureAWSPackage.
5. In the **Select Targets by** section, choose an option.

   **Note**
   
   If you choose to target instances by using tags, and you specify tags that map to Linux instances, the association succeeds on the Windows instance, but fails on the Linux instances. The overall status of the association shows Failed.

6. In the **Schedule** section, choose an option. Updated PV drivers are only released a few times a year, so you can schedule the association to run once a month, if you want.
7. In the **Parameters** section, choose Install from the **Action** list.
8. For **Name** list, enter AWSPVDriver. You can leave the **Version** field empty.
9. In the **Advanced** section, choose Write to S3 if you want to write association details to an Amazon S3 bucket.
10. Disregard the **S3Region** field. This field is deprecated. Specify the name of your bucket in the **S3Bucket Name** field. If you want to write output to a sub-folder, specify the sub-folder name in the **S3Key Prefix** field.

11. Choose **Create Association**, and then choose **Close**. The system attempts to create the association on the instance(s) and immediately apply the state. The association status shows **Pending**.

12. In the right corner of the **Association** page, choose the refresh button. If you created the association on one or more EC2 instances for Windows Server, the status changes to **Success**. If your instances are not properly configured for Systems Manager, or if you inadvertently targeted Linux instances, the status shows **Failed**.

13. If the status is **Failed**, choose the **Instances** tab and verify that the association was successfully created on your EC2 instances for Windows Server. If EC2 instances for Windows Server show a status of **Failed**, verify that SSM Agent is running on the instance, and verify that the instance is configured with an IAM role for Systems Manager. For more information, see Systems Manager prerequisites (p. 14).

### AWS Systems Manager Patch Manager

AWS Systems Manager Patch Manager automates the process of patching managed instances with both security related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for Microsoft applications.) You can use Patch Manager to install Service Packs on Windows instances and perform minor version upgrades on Linux instances. You can patch fleets of EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Windows Server, Amazon Linux, Amazon Linux 2, CentOS, Debian, Oracle Linux, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), and Ubuntu Server. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.

**Important**

AWS does not test patches for Windows Server or Linux before making them available in Patch Manager. Also, Patch Manager doesn’t support upgrading major versions of operating systems, such as Windows Server 2016 to Windows Server 2019, or SUSE Linux Enterprise Server (SLES) 12.0 to SLES 15.0.

Patch Manager uses *patch baselines*, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager maintenance window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags. (Tags are keys that help identify and sort your resources within your organization.) You can add tags to your patch baselines themselves when you create or update them.

Patch Manager integrates with AWS Identity and Access Management (IAM), AWS CloudTrail, and Amazon CloudWatch Events to provide a secure patching experience that includes event notifications and the ability to audit usage.

For information about using CloudTrail to monitor Systems Manager actions, see Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144).

For information about using CloudWatch Events to monitor Systems Manager events, see Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147).

### Getting Started with Patch Manager

To get started with Patch Manager, complete the tasks described in the following table.

<table>
<thead>
<tr>
<th>Task</th>
<th>For more information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify Systems Manager prerequisites</td>
<td>Systems Manager prerequisites (p. 14)</td>
</tr>
</tbody>
</table>
Task | For more information
--- | ---
Learn how to set up and configure patching | Working with Patch Manager (console) (p. 958)
Configure permissions for Maintenance Windows (Required if you intend to use this feature when patching.) | Controlling access to maintenance windows (p. 621)
Create patch baselines, patch groups, and a maintenance window to run patching in a test environment | Working with Patch Manager (console) (p. 958)

Topics
- Patch Manager prerequisites (p. 912)
- How Patch Manager operations work (p. 913)
- About patching operations (p. 935)
- About patch baselines (p. 948)
- Working with Patch Manager (console) (p. 958)
- Working with Patch Manager (AWS CLI) (p. 973)
- AWS Systems Manager Patch Manager walkthroughs (p. 990)

Patch Manager prerequisites

SSM Agent Version

Version 2.0.834.0 or later of SSM Agent be running on the instances you want to manage with Patch Manager.

**Note**
An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Supported Operating Systems

The Patch Manager capability does not support all the same operating systems versions that are supported by other AWS Systems Manager capabilities. For example, Patch Manager does not support CentOS 6.3 or Raspbian Stretch. (For the full list of Systems Manager-supported operating systems, see Systems Manager prerequisites (p. 14).) Therefore, ensure that the instances you want to use with Patch Manager are running one of the operating systems listed in the following table.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Details</th>
</tr>
</thead>
</table>
| Linux            | • Amazon Linux 2012.03 - 2018.03  
                  | • Amazon Linux 2 2 - 2.0 
                  | • CentOS 6.5 - 7.7 
                  | • Debian 8.x and 9.x 
                  | • Oracle Linux 7.6 
                  | • Red Hat Enterprise Linux (RHEL) 6.5 - 8.2 |
### How it works

#### Operating system

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSE Linux Enterprise Server (SLES) 12.0 and later 12.x versions, 15.0 and 15.1</td>
<td></td>
</tr>
<tr>
<td>Ubuntu Server 14.04 LTS, 16.04 LTS, and 18.04 LTS</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

Instances created from an Amazon Linux AMI that are using a proxy must be running a current version of the Python `requests` module in order to support Patch Manager operations. For more information, see Upgrade the Python `requests` module on Amazon Linux instances that use a proxy server (p. 88).


---

### How Patch Manager operations work

This section provides technical details that explain how Patch Manager determines which patches to install and how it installs them on each supported operating system. For Linux operating systems, it also provides information about specifying a source repository, in a custom patch baseline, for patches other than the default configured on an instance. This section also provides details about how patch baseline rules work on different distributions of the Linux operating system.

**Topics**

- How security patches are selected (p. 913)
- How to specify an alternative patch source repository (Linux) (p. 917)
- How patches are installed (p. 919)
- How patch baseline rules work on Linux-based systems (p. 924)
- Key differences between Linux and Windows patching (p. 933)

### How security patches are selected

The primary focus of Patch Manager is on installing operating systems security-related updates on instances. By default, Patch Manager doesn't install all available patches, but rather a smaller set of patches focused on security.

**Note**

On all Linux-based systems supported by Patch Manager, you can choose a different source repository configured for the instance, typically to install nonsecurity updates. For information, see How to specify an alternative patch source repository (Linux) (p. 917).

The remainder of this section explains how Patch Manager selects security patches for the different supported operating systems.

**Amazon Linux and Amazon Linux 2**

On Amazon Linux and Amazon Linux 2, the Systems Manager patch baseline service uses preconfigured repositories on the instance. There are usually two preconfigured repositories (repos) on an instance:
• **Repo ID**: amzn-main/latest  
  **Repo name**: amzn-main-Base

• **Repo ID**: amzn-updates/latest  
  **Repo name**: amzn-updates-Base

**Note**  
All updates are downloaded from the remote repos configured on the instance. Therefore, the instance must be able to connect to the repos so the patching can be performed.

Amazon Linux and Amazon Linux 2 instances use Yum as the package manager, and Yum uses the concept of an update notice as a file named `updateinfo.xml`. An update notice is simply a collection of packages that fix specific problems. All packages that are in an update notice are considered Security by Patch Manager. Individual packages are not assigned classifications or severity levels. For this reason, Patch Manager assigns the attributes of an update notice to the related packages.

**Note**  
If you select the **Approved patches include non-security updates** check box in the Create patch baseline page, then packages that are not classified in an `updateinfo.xml` file (or a package that contains a file without properly formatted Classification, Severity, and Date values) can be included in the prefiltered list of patches. However, in order for a patch to be applied, the patch must still meet the user-specified patch baseline rules.

**CentOS**

On CentOS, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the instance. Here are some examples from a CentOS 6.9 Amazon Machine Image (AMI):

• **Repo ID**: ultra-centos-6.9-base  
  **Repo name**: UltraServe CentOS-6.9 - Base

• **Repo ID**: ultra-centos-6.9-extras  
  **Repo name**: UltraServe CentOS-6.9 - Extras

• **Repo ID**: ultra-centos-6.9-updates  
  **Repo name**: UltraServe CentOS-6.9 - Updates

• **Repo ID**: ultra-centos-6.x-glusterfs  
  **Repo name**: UltraServe CentOS-6.x - GlusterFS

• **Repo ID**: ultra-centos-6.x-ultrarepo  
  **Repo name**: UltraServe CentOS-6.x – UltraServe Repo Packages

**Note**  
All updates are downloaded from the remote repos configured on the instance. Therefore, the instance must be able to connect to the repos so the patching can be performed.

CentOS instances use Yum as the package manager, and Yum uses the concept of an update notice. An update notice is simply a collection of packages that fix specific problems. All packages that are in an update notice are considered Security packages by Patch Manager. However, CentOS default repos aren’t configured with an update notice. This means that Patch Manager does not detect packages on a default CentOS repo. To enable Patch Manager to process packages that aren’t contained in an update notice, you must enable the **EnableNonSecurity** flag in the patch baseline rules.
### Note
CentOS update notices are supported. Repos with update notices can be downloaded after launch.

#### Debian

On Debian, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the instance. These preconfigured repos are used to pull an updated list of available package upgrades. For this, Systems Manager performs the equivalent of a `sudo apt-get update` command.

Packages are then filtered from `debian-security codename` repos, where the codename is something like `jessie` or `stretch`. For example, on Debian 8, Patch Manager only identifies upgrades that are part of `debian-security jessie`. On Debian 9, only upgrades that are part of `debian-security stretch` are identified.

#### Oracle Linux

On Oracle Linux, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the instance. There are usually two preconfigured repos on an instance:

- **Repo ID**: ol7_UEKR5/x86_64
  - **Repo name**: Latest Unbreakable Enterprise Kernel Release 5 for Oracle Linux 7 Server (x86_64)
- **Repo ID**: ol7_latest/x86_64
  - **Repo name**: Oracle Linux 7 Server Latest (x86_64)

#### RHEL

On Red Hat Enterprise Linux, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the instance. There are usually three preconfigured repos on an instance.

All updates are downloaded from the remote repos configured on the instance. Therefore, the instance must be able to connect to the repos so the patching can be performed.

### Note

Oracle Linux instances use Yum as the package manager, and Yum uses the concept of an update notice as a file named `updateinfo.xml`. An update notice is simply a collection of packages that fix specific problems. All packages that are in an update notice are considered Security by Patch Manager. Individual packages are not assigned classifications or severity levels. For this reason, Patch Manager assigns the attributes of an update notice to the related packages.

#### Note

If you select the **Approved patches include non-security updates** check box in the **Create patch baseline** page, then packages that are not classified in an `updateinfo.xml` file (or a package that contains a file without properly formatted Classification, Severity, and Date values) can be included in the prefiltered list of patches. However, in order for a patch to be applied, the patch must still meet the user-specified patch baseline rules.
How it works

A package that contains a file without properly formatted Classification, Severity, and Date values can be included in the prefiltered list of patches. However, in order for a patch to be applied, the patch must still meet the user-specified patch baseline rules.

Red Hat Enterprise Linux 7 instances use Yum as the package manager. Red Hat Enterprise Linux 8 instances use DNF as the package manager. Both package managers use the concept of an update notice as a file named `updateinfo.xml`. An update notice is simply a collection of packages that fix specific problems. All packages that are in an update notice are considered Security by Patch Manager. Individual packages are not assigned classifications or severity levels. For this reason, Patch Manager assigns the attributes of an update notice to the related packages.

Note that repo locations differ between RHEL 7 and RHEL 8:

### RHEL 7

**Note**
The following repo IDs are associated with RHUI 2. RHUI 3 launched in December 2019 and introduced a different naming scheme for Yum repository IDs. Depending on the RHEL-7 AMI you create your instances from, you might need to update your commands. For more information, see [Repository IDs for RHEL 7 in AWS Have Changed](https://access.redhat.com/site/en/articles/15750) on the Red Hat Customer Portal.

- **Repo ID**: rhui-REGION-client-config-server-7/x86_64
- **Repo name**: Red Hat Update Infrastructure 2.0 Client Configuration Server 7
- **Repo ID**: rhui-REGION-rhel-server-releases/7Server/x86_64
- **Repo name**: Red Hat Enterprise Linux Server 7 (RPMs)
- **Repo ID**: rhui-REGION-rhel-server-rh-common/7Server/x86_64
- **Repo name**: Red Hat Enterprise Linux Server 7 RH Common (RPMs)

### RHEL 8

- **Repo ID**: rhel-8-appstream-rhui-rpms
- **Repo name**: Red Hat Enterprise Linux 8 for x86_64 - AppStream from RHUI (RPMs)
- **Repo ID**: rhel-8-baseos-rhui-rpms
- **Repo name**: Red Hat Enterprise Linux 8 for x86_64 - BaseOS from RHUI (RPMs)
- **Repo ID**: rhui-client-config-server-8
- **Repo name**: Red Hat Update Infrastructure 3 Client Configuration Server 8

### SLES

On SUSE Linux Enterprise Server (SLES) instances, the ZYPP library gets the list of available patches (a collection of packages) from the following locations:

- List of repositories: `etc/zypp/repos.d/*`
- Package information: `/var/cache/zypp/raw/*`

SLES instances use Zypper as the package manager, and Zypper uses the concept of a patch. A patch is simply a collection of packages that fix a specific problem. Patch Manager handles all packages referenced in a patch as security-related. Because individual packages aren't given classifications or severity, Patch Manager assigns the packages the attributes of the patch that they belong to.

### Ubuntu Server

On Ubuntu Server, the Systems Manager patch baseline service uses preconfigured repositories (repos) on the instance. These preconfigured repos are used to pull an updated list of available packages.
package upgrades. For this, Systems Manager performs the equivalent of a `sudo apt-get update` command.

Packages are then filtered from `codename-security` repos, where the codename is something like `trusty` or `xenial`. For example, on Ubuntu Server 14, Patch Manager only identifies upgrades that are part of `trusty-security`. On Ubuntu Server 16, only upgrades that are part of `xenial-security` are identified.

**Windows**

On Microsoft Windows operating systems, Patch Manager retrieves a list of available updates that Microsoft publishes through its update services, such as Microsoft Update and Windows Server Update Services (WSUS). Patch Manager continuously monitors for new updates in every AWS Region. The list of available updates is refreshed in each Region at least once per day. When the patch information from Microsoft is processed, Patch Manager removes updates that have been replaced by later updates from its patch list. Therefore, only the most recent update is displayed and made available for installation. For example, if `KB4012214` replaces `KB3135456`, only `KB4012214` is made available as an update in Patch Manager.

**Note**
Patch Manager only makes available patches for Windows Server operating system versions that are supported for Patch Manager. For example, Patch Manager can't be used to patch Windows RT.

**How to specify an alternative patch source repository (Linux)**

When you use the default repositories configured on an instance for patching operations, Patch Manager scans for or installs security-related patches. This is the default behavior for Patch Manager. For complete information on how Patch Manager selects and installs security patches, see How security patches are selected (p. 913).

On Linux systems, however, you can also use Patch Manager to install patches that are not related to security, or that are in a different source repository than the default one configured on the instance. You can specify alternative patch source repositories when you create a custom patch baseline. In each custom patch baseline, you can specify patch source configurations for up to 20 versions of a supported Linux operating system.

For example, suppose that your Ubuntu Server fleet includes both Ubuntu Server 14.04 and Ubuntu Server 16.04 instances. In this case, you can specify alternate repositories for each version in the same custom patch baseline. For each version, you provide a name, specify the operating system version type (product), and provide a repository configuration. You can also specify a single alternative source repository that applies to all versions of a supported operating system.

**Note**
Running a custom patch baseline that specifies alternative patch repositories on an instance doesn't change the default repository configured for the instance.

For a list of example scenarios for using this option, see Sample uses for alternative patch source repositories (p. 918) later in this topic.

For information about default and custom patch baselines, see About predefined and custom patch baselines (p. 948).

**Using the Console**

To specify alternative patch source repositories when you are working in the AWS Systems Manager console, use the Patch sources section on the Create patch baseline page. For information about using the Patch sources options, see Create a custom patch baseline (Linux) (p. 962).

**Using the AWS CLI**
For an example of using the `--sources` option with the CLI, see Create a patch baseline with custom repositories for different OS versions (p. 975).

**Topics**
- Important considerations for alternative repositories (p. 918)
- Sample uses for alternative patch source repositories (p. 918)

**Important considerations for alternative repositories**

Keep in mind the following points as you plan your patching strategy using alternative patch repositories.

**Only specified repositories are used for patching**

Specifying alternative repositories doesn't mean specifying additional repositories. You can choose to specify repositories other than those configured as defaults on an instance. However, you must also specify the default repositories as part of the alternative patch source configuration if you want their updates to be applied.

For example, on Amazon Linux 2 instances, the default repositories are `amzn-main` and `amzn-update`. If you want to include the Extra Packages for Enterprise Linux (EPEL) repository in your patching operations, you must specify all three repositories as alternative repositories.

**Note**

Running a custom patch baseline that specifies alternative patch repositories on an instance doesn't change the default repository configured for the instance.

**Patching behavior for YUM-based distributions depends on the updateinfo.xml manifest**

When you specify alternative patch repositories for YUM-based distributions, such as Amazon Linux or Amazon Linux 2, Red Hat Enterprise Linux, or CentOS, patching behavior depends on whether the repository includes an update manifest in the form of a complete and correctly formatted `updateinfo.xml` file. This file specifies the release date, classifications, and severities of the various packages. Any of the following will affect the patching behavior:

- **If you filter on Classification and Severity**, but they aren't specified in `updateinfo.xml`, the package will not be included by the filter. This also means that packages without an `updateinfo.xml` file won't be included in patching.

- **If you filter on ApprovalAfterDays**, but the package release date isn't in Unix Epoch format (or has no release date specified), the package will not be included by the filter.

- **There is an exception if you select the Approved patches include non-security updates check box** in the Create patch baseline page. In this case, packages without an `updateinfo.xml` file (or that contains this file without properly formatted Classification, Severity, and Date values) will be included in the prefiltered list of patches. (They must still meet the other patch baseline rule requirements in order to be installed.)

**Sample uses for alternative patch source repositories**

**Example 1 – Nonsecurity Updates for Ubuntu Server**

You are already using Patch Manager to install security patches on a fleet of Ubuntu Server instances using the AWS-provided predefined patch baseline **AWS-UbuntuDefaultPatchBaseline**. You can create a new patch baseline that is based on this default, but specify in the approval rules that you want nonsecurity related updates that are part of the default distribution to be installed as well. When this patch baseline is run against your instances, patches for both security and nonsecurity issues are applied. You can also choose to approve nonsecurity patches in the patch exceptions you specify for a baseline.
Your Ubuntu Server instances are running software that is distributed through a Personal Package Archives (PPA) for Ubuntu. In this case, you create a patch baseline that specifies a PPA repository that you have configured on the instance as the source repository for the patching operation. Then use Run Command to run the patch baseline document on the instances.

Example 3 – Internal Corporate Applications on Amazon Linux

You need to run some applications needed for industry regulatory compliance on your Amazon Linux instances. You can configure a repository for these applications on the instances, use YUM to initially install the applications, and then update or create a new patch baseline to include this new corporate repository. After this you can use Run Command to run the AWS-RunPatchBaseline document with the Scan option to see if the corporate package is listed among the installed packages and is up to date on the instance. If it isn't up to date, you can run the document again using the Install option to update the applications.

How patches are installed

Patch Manager uses the appropriate built-in mechanism for an operating system type to install updates on an instance. For example, on Windows Server, the Windows Update API is used, and on Amazon Linux the yum package manager is used.

The remainder of this section explains how Patch Manager installs patches on an operating system.

Amazon Linux and Amazon Linux 2

On Amazon Linux and Amazon Linux 2 instances, the patch installation workflow is as follows:

1. Apply GlobalFilters as specified in the patch baseline, keeping only the qualified packages for further processing.
2. Apply ApprovalRules as specified in the patch baseline. Each approval rule can define a package as approved.
3. Apply ApprovedPatches as specified in the patch baseline. The approved patches are approved for update even if they are discarded by GlobalFilters or if no approval rule specified in ApprovalRules grants it approval.
4. Apply RejectedPatches as specified in the patch baseline. The rejected patches are removed from the list of approved patches and will not be applied.
5. If multiple versions of a patch are approved, the latest version is applied.
6. The YUM update API is applied to approved patches as follows:
   • For predefined default patch baselines provided by AWS, and for custom patch baselines where the Approved patches include non-security updates check box is not selected, only patches specified in updateinfo.xml are applied (security updates only).
   • For custom patch baselines where the Approved patches include non-security updates is selected, both patches in updateinfo.xml and those not in updateinfo.xml are applied (security and nonsecurity updates).

   The equivalent yum command for this workflow is:
   
   ```bash
   sudo yum update-minimal --sec-severity=critical,important --bugfix -y
   ```

   ```bash
   sudo yum update --security --bugfix
   ```

7. The instance is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance
is not rebooted after Patch Manager runs. For more information, see Parameter name: RebootOption (p. 945).)

CentOS

On CentOS instances, the patch installation workflow is as follows:

1. Apply GlobalFilters as specified in the patch baseline, keeping only the qualified packages for further processing.
2. Apply ApprovalRules as specified in the patch baseline. Each approval rule can define a package as approved.
3. Apply ApprovedPatches as specified in the patch baseline. The approved patches are approved for update even if they are discarded by GlobalFilters or if no approval rule specified in ApprovalRules grants it approval.
4. Apply RejectedPatches as specified in the patch baseline. The rejected patches are removed from the list of approved patches and will not be applied.
5. If multiple versions of a patch are approved, the latest version is applied.
6. The YUM update API is applied to approved patches.
7. The instance is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: RebootOption (p. 945).)

Debian

On Debian instances, the patch installation workflow is as follows:

1. On Debian 8 only: Because some Debian 8.* instances refer to an obsolete package repository (jessie-backports), Patch Manager performs the following additional steps to ensure that patching operations succeed:
   a. On your instance, the reference to the jessie-backports repository is commented out from the source location list (/etc/apt/sources.list.d/jessie-backports). As a result, no attempt is made to download patches from that location.
   b. A Stretch security update signing key is imported. This key provides the necessary permissions for the update and install operations on Debian 8.* distributions.
   c. An apt-get operation is run to ensure that the latest version of python3-apt is installed before the patching process begins.
   d. After the installation process completes, the reference to the jessie-backports repository is restored and the signing key is removed from the apt sources keyring. This is done to leave the system configuration as it was before the patching operation.

The next time Patch Manager updates the system, the same process is repeated.

The remaining steps in the patch installation workflow apply to both Debian 8 and 9.
2. Apply GlobalFilters as specified in the patch baseline, keeping only the qualified packages for further processing.
3. Apply ApprovalRules as specified in the patch baseline. Each approval rule can define a package as approved. In addition, an implicit rule is applied in order to select only packages with upgrades in security repos. For each package, the candidate version of the package (which is typically the latest version) must be part of a security repo.

   Note
   Because it's not possible to reliably determine the release dates of update packages for Debian, the auto-approval options are not supported for this operating system.
4. Apply **ApprovedPatches** as specified in the patch baseline. The approved patches are approved for update even if they are discarded by **GlobalFilters** or if no approval rule specified in **ApprovalRules** grants it approval.

5. Apply **RejectedPatches** as specified in the patch baseline. The rejected patches are removed from the list of approved patches and will not be applied.

6. The APT library is used to upgrade packages.

7. The instance is rebooted if any updates were installed. (Exception: If the **RebootOption** parameter is set to **NoReboot** in the AWS–RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: **RebootOption** (p. 945).)

**Oracle Linux**

On Oracle Linux instances, the patch installation workflow is as follows:

1. Apply **GlobalFilters** as specified in the patch baseline, keeping only the qualified packages for further processing.

2. Apply **ApprovalRules** as specified in the patch baseline. Each approval rule can define a package as approved.

3. Apply **ApprovedPatches** as specified in the patch baseline. The approved patches are approved for update even if they are discarded by **GlobalFilters** or if no approval rule specified in **ApprovalRules** grants it approval.

4. Apply **RejectedPatches** as specified in the patch baseline. The rejected patches are removed from the list of approved patches and will not be applied.

5. If multiple versions of a patch are approved, the latest version is applied.

6. The YUM update API is applied to approved patches as follows:

   - For predefined default patch baselines provided by AWS, and for custom patch baselines where the **Approved patches include non-security updates** check box is not selected, only patches specified in updateinfo.xml are applied (security updates only).

   The equivalent yum command for this workflow is:
   ```
   sudo yum update-minimal --sec-severity=important,moderate --bugfix -y
   ```

   - For custom patch baselines where the **Approved patches include non-security updates** is selected, both patches in updateinfo.xml and those not in updateinfo.xml are applied (security and nonsecurity updates).

   The equivalent yum command for this workflow is:
   ```
   sudo yum update --security --bugfix -y
   ```

7. The instance is rebooted if any updates were installed. (Exception: If the **RebootOption** parameter is set to **NoReboot** in the AWS–RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: **RebootOption** (p. 945).)

**RHEL**

On Red Hat Enterprise Linux instances, the patch installation workflow is as follows:

1. Apply **GlobalFilters** as specified in the patch baseline, keeping only the qualified packages for further processing.
2. Apply **ApprovalRules** as specified in the patch baseline. Each approval rule can define a package as approved.

3. Apply **ApprovedPatches** as specified in the patch baseline. The approved patches are approved for update even if they are discarded by **GlobalFilters** or if no approval rule specified in **ApprovalRules** grants it approval.

4. Apply **RejectedPatches** as specified in the patch baseline. The rejected patches are removed from the list of approved patches and will not be applied.

5. If multiple versions of a patch are approved, the latest version is applied.

6. The YUM update API (on RHEL 7) or the DNF update API (on RHEL 8) is applied to approved patches as follows:

   - For predefined default patch baselines provided by AWS, and for custom patch baselines where the **Approved patches include non-security updates** check box is not selected, only patches specified in `updateinfo.xml` are applied (security updates only).

     For RHEL 7, the equivalent `yum` command for this workflow is:

     ```bash
     sudo yum update-minimal --sec-severity=critical,important --bugfix -y
     ```

     For RHEL 8, the equivalent `dnf` commands for this workflow is:

     ```bash
     sudo dnf update-minimal --sec-severity=Critical --bugfix -y ;
     sudo dnf update-minimal --sec-severity=Important --bugfix -y
     ```

   - For custom patch baselines where the **Approved patches include non-security updates** is selected, both patches in `updateinfo.xml` and those not in `updateinfo.xml` are applied (security and nonsecurity updates).

     For RHEL 7, the equivalent `yum` command for this workflow is:

     ```bash
     sudo yum update --security --bugfix -y
     ```

     For RHEL 8, the equivalent `yum` command for this workflow is:

     ```bash
     sudo dnf update --security --bugfix -y
     ```

7. The instance is rebooted if any updates were installed. (Exception: If the **RebootOption** parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: **RebootOption** (p. 945).)

**SLES**

On SUSE Linux Enterprise Server (SLES) instances, the patch installation workflow is as follows:

1. Apply **GlobalFilters** as specified in the patch baseline, keeping only the qualified packages for further processing.

2. Apply **ApprovalRules** as specified in the patch baseline. Each approval rule can define a package as approved.

3. Apply **ApprovedPatches** as specified in the patch baseline. The approved patches are approved for update even if they are discarded by **GlobalFilters** or if no approval rule specified in **ApprovalRules** grants it approval.

4. Apply **RejectedPatches** as specified in the patch baseline. The rejected patches are removed from the list of approved patches and won't be applied.

5. If multiple versions of a patch are approved, the latest version is applied.
6. The Zypper update API is applied to approved patches.

7. The instance is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: RebootOption (p. 945).)

**Ubuntu Server**

On Ubuntu Server instances, the patch installation workflow is as follows:

1. Apply GlobalFilters as specified in the patch baseline, keeping only the qualified packages for further processing.

2. Apply ApprovalRules as specified in the patch baseline. Each approval rule can define a package as approved. In addition, an implicit rule is applied in order to select only packages with upgrades in security repos. For each package, the candidate version of the package (which is typically the latest version) must be part of a security repo.

   **Note**
   Because it's not possible to reliably determine the release dates of update packages for Ubuntu Server, the auto-approval options are not supported for this operating system.

3. Apply ApprovedPatches as specified in the patch baseline. The approved patches are approved for update even if they are discarded by GlobalFilters or if no approval rule specified in ApprovalRules grants it approval.

4. Apply RejectedPatches as specified in the patch baseline. The rejected patches are removed from the list of approved patches and will not be applied.

5. The APT library is used to upgrade packages.

6. The instance is rebooted if any updates were installed. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: RebootOption (p. 945).)

**Windows**

When a patching operation is performed on a Windows Server instance, the instance requests a snapshot of the appropriate patch baseline from Systems Manager. This snapshot contains the list of all updates available in the patch baseline that have been approved for deployment. This list of updates is sent to the Windows Update API, which determines which of the updates are applicable to the instance and installs them as needed. If any updates are installed, the instance is rebooted afterwards, as many times as necessary to complete all necessary patching. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: RebootOption (p. 945).) The summary of the patching operation can be found in the output of the Run Command request. Additional logs can be found on the instance in the %PROGRAMDATA%\Amazon\PatchBaselineOperations\Logs folder.

Because the Windows Update API is used to download and install patches, all Group Policy settings for Windows Update are respected. No Group Policy settings are required to use Patch Manager, but any settings that you have defined will be applied, such as to direct instances to a Windows Server Update Services (WSUS) server.

   **Note**
   By default, Windows downloads all patches from Microsoft's Windows Update site because Patch Manager uses the Windows Update API to drive the download and installation of patches. As a result, the instance must be able to reach the Microsoft Windows Update site or patching will fail. Alternatively, you can configure a WSUS server to serve as a patch
repository and configure your instances to target that WSUS server instead using Group Policies.

How patch baseline rules work on Linux-based systems

The rules in a patch baseline for Linux distributions operate differently based on the distribution type. Unlike patch updates on Windows Server instances, rules are evaluated on each instance to take the configured repos on the instance into consideration. Patch Manager uses the native package manager to drive the installation of patches approved by the patch baseline.

Topics

• How patch baseline rules work on Amazon Linux and Amazon Linux 2 (p. 924)
• How patch baseline rules work on CentOS (p. 926)
• How patch baseline rules work on Debian (p. 928)
• How patch baseline rules work on Oracle Linux (p. 928)
• How patch baseline rules work on RHEL (p. 930)
• How patch baseline rules work on SUSE Linux Enterprise Server (p. 932)
• How patch baseline rules work on Ubuntu Server (p. 933)

How patch baseline rules work on Amazon Linux and Amazon Linux 2

On Amazon Linux and Amazon Linux 2, the patch selection process is as follows:

1. On the instance, the YUM library accesses the updateinfo.xml file for each configured repo.

   **Note**
   
   If no updateinfo.xml file is found, no patch will be applied.

2. Each update notice in updateinfo.xml includes several attributes that denote the properties of the packages in the notice, as described in the following table.

Update notice attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>Corresponds to the value of the Classification key attribute in the patch baseline's PatchFilter data type. Denotes the type of package included in the update notice. You can view the list of supported values by using the AWS CLI command <code>describe-patch-properties</code> or the API action <code>DescribePatchProperties</code>. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console.</td>
</tr>
<tr>
<td><strong>severity</strong></td>
<td>Corresponds to the value of the Severity key attribute patch baseline's PatchFilter data type. Denotes the severity of the packages included in the update notice. Usually only applicable for Security update notices. You can view the list of supported values by using the AWS CLI command <code>describe-patch-properties</code> or the API action <code>DescribePatchProperties</code>.</td>
</tr>
</tbody>
</table>

924
### Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>patch-properties</strong> or the API action DescribePatchProperties. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console.</td>
<td></td>
</tr>
<tr>
<td><strong>update_id</strong></td>
<td>Denotes the advisory ID, such as ALAS-2017-867. The advisory ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.</td>
</tr>
<tr>
<td><strong>references</strong></td>
<td>Contains additional information about the update notice, such as a CVE ID (format: CVE-2017-1234567). The CVE ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.</td>
</tr>
<tr>
<td><strong>updated</strong></td>
<td>Corresponds to ApproveAfterDays in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the ApproveAfterDays is used to determine if the patch is approved for deployment.</td>
</tr>
</tbody>
</table>

**Note**

For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

3. The product of the instance is determined by SSM Agent. This attribute corresponds to the value of the Product key attribute in the patch baseline's PatchFilter data type.

4. Packages are selected for the update according to the following guidelines:

<table>
<thead>
<tr>
<th>Security option</th>
<th>Patch selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-defined default patch baselines provided by AWS and custom patch baselines where the <strong>Approved patches include non-security updates</strong> is not selected</td>
<td>For each update notice in updateinfo.xml, the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used.</td>
</tr>
<tr>
<td></td>
<td>The equivalent yum command for this workflow is:</td>
</tr>
<tr>
<td></td>
<td>sudo yum update-minimal --sec-severity=critical,important --bugfix -y</td>
</tr>
<tr>
<td>Custom patch baselines where the <strong>Approved patches include non-security updates</strong> is selected</td>
<td>In addition to applying the security updates that have been selected from updateinfo.xml, Patch Manager will apply nonsecurity updates that otherwise meet the patch filtering rules.</td>
</tr>
</tbody>
</table>
Security option | Patch selection
--- | ---
 | The equivalent yum command for this workflow is:
 | sudo yum update --security --bugfix -y

For information about patch compliance status values, see About patch compliance status values (p. 946).

How patch baseline rules work on CentOS

On CentOS, the patch selection process is as follows:

1. On the instance, the YUM library accesses the updateinfo.xml file for each configured repo.

   **Note**
   
   The updateinfo.xml file might not be available if the repo is not one managed by Oracle. If there is no updateinfo.xml found, no patch will be applied.

2. Each update notice in updateinfo.xml includes several attributes that denote the properties of the packages in the notice, as described in the following table.

### Update notice attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Corresponds to the value of the Classification key attribute in the patch baseline's PatchFilter data type. Denotes the type of package included in the update notice. You can view the list of supported values by using the AWS CLI command describe-patch-properties or the API action DescribePatchProperties. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console.</td>
</tr>
<tr>
<td>severity</td>
<td>Corresponds to the value of the Severity key attribute in the patch baseline's PatchFilter data type. Denotes the severity of the packages included in the update notice. Usually only applicable for Security update notices. You can view the list of supported values by using the AWS CLI command describe-patch-properties or the API action DescribePatchProperties. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console.</td>
</tr>
<tr>
<td>update_id</td>
<td>Denotes the advisory ID, such as CVE-2019-17055. The advisory ID can be used</td>
</tr>
</tbody>
</table>
### How it works

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>references</td>
<td>Contains additional information about the update notice, such as a CVE ID (format: CVE-2019-17055) or a Bugzilla ID (format: 1463241). The CVE ID and Bugzilla ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.</td>
</tr>
<tr>
<td>updated</td>
<td>Corresponds to ApproveAfterDays in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the ApproveAfterDays is used to determine if the patch is approved for deployment.</td>
</tr>
</tbody>
</table>

**Note**

For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

3. The product of the instance is determined by SSM Agent. This attribute corresponds to the value of the Product key attribute in the patch baseline's PatchFilter data type.

4. Packages are selected for the update according to the following guidelines:

<table>
<thead>
<tr>
<th>Security option</th>
<th>Patch selection</th>
</tr>
</thead>
</table>
| Pre-defined default patch baselines provided by AWS and custom patch baselines where the **Approved patches include non-security updates** is **not** selected | For each update notice in updateinfo.xml, the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used. The equivalent yum command for this workflow is: 
```bash
sudo yum update-minimal --sec-severity=critical,important --bugfix -y
```

| Custom patch baselines where the **Approved patches include non-security updates** is **selected** | In addition to applying the security updates that have been selected from updateinfo.xml, Patch Manager will apply nonsecurity updates that otherwise meet the patch filtering rules. The equivalent yum command for this workflow is: 
```bash
sudo yum update --security --bugfix -y
```
For information about patch compliance status values, see [About patch compliance status values](p. 946).

### How patch baseline rules work on Debian

On Debian, the patch baseline service offers filtering on the *Priority* and *Section* fields. These fields are typically present for all Debian packages. To determine whether a patch is selected by the patch baseline, Patch Manager does the following:

1. On Debian systems, the equivalent of `sudo apt-get update` is run to refresh the list of available packages. Repos are not configured and the data is pulled from repos configured in a `sources` list.

   **Important**
   - On Debian 8 only: Because Debian 8.* operating systems refer to an obsolete package repository (`jessie-backports`), Patch Manager performs the following additional steps to ensure that patching operations succeed:
     a. On your instance, the reference to the `jessie-backports` repository is commented out from the source location list (`/etc/apt/sources.list.d/jessie-backports`). As a result, no attempt is made to download patches from that location.
     b. A Stretch security update signing key is imported. This key provides the necessary permissions for the update and install operations on Debian 8.* distributions.
     c. An `apt-get` operation is run to ensure that the latest version of `python3-apt` is installed before the patching process begins.
     d. After the installation process completes, the reference to the `jessie-backports` repository is restored and the signing key is removed from the apt sources keyring. This is done to leave the system configuration as it was before the patching operation.

2. Next, the `GlobalFilters`, `ApprovalRules`, `ApprovedPatches` and `RejectedPatches` lists are applied.

   **Note**
   - Because it’s not possible to reliably determine the release dates of update packages for Debian, the auto-approval options are not supported for this operating system.

   Only packages with candidate versions appearing in the distribution security repo (archive) are selected. For Debian 8 this is repo is `debian-security jessie`. For Debian 9, it is `debian-security stretch`.

   **Note**
   - For information about accepted formats for lists of approved patches and rejected patches, see [About package name formats for approved and rejected patch lists](p. 952).

To view the contents of the *Priority* and *Section* fields, run the following `aptitude` command:

**Note**
- You may need to first install Aptitude on Debian systems.

```
aptitude search -F '%p %P %s %t %V#' '~U'
```

In the response to this command, all upgradable packages are reported in this format:

```
nname, priority, section, archive, candidate version
```

For information about patch compliance status values, see [About patch compliance status values](p. 946).

### How patch baseline rules work on Oracle Linux

On Oracle Linux, the patch selection process is as follows:
1. On the instance, the YUM library accesses the `updateinfo.xml` file for each configured repo.

   **Note**
   The `updateinfo.xml` file might not be available if the repo is not one managed by Oracle. If there is no `updateinfo.xml` found, no patch will be applied.

2. Each update notice in `updateinfo.xml` includes several attributes that denote the properties of the packages in the notice, as described in the following table.

   **Update notice attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Corresponds to the value of the Classification key attribute in the patch baseline's <code>PatchFilter</code> data type. Denotes the type of package included in the update notice. You can view the list of supported values by using the AWS CLI command <code>describe-patch-properties</code> or the API action <code>DescribePatchProperties</code>. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console.</td>
</tr>
<tr>
<td>severity</td>
<td>Corresponds to the value of the Severity key attribute in the patch baseline's <code>PatchFilter</code> data type. Denotes the severity of the packages included in the update notice. Usually only applicable for Security update notices. You can view the list of supported values by using the AWS CLI command <code>describe-patch-properties</code> or the API action <code>DescribePatchProperties</code>. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console.</td>
</tr>
<tr>
<td>update_id</td>
<td>Denotes the advisory ID, such as <code>CVE-2019-17055</code>. The advisory ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.</td>
</tr>
<tr>
<td>references</td>
<td>Contains additional information about the update notice, such as a CVE ID (format: <code>CVE-2019-17055</code>) or a Bugzilla ID (format: <code>1463247</code>). The CVE ID and Bugzilla ID can be used in the ApprovedPatches or RejectedPatches attribute in the patch baseline.</td>
</tr>
<tr>
<td>updated</td>
<td>Corresponds to ApproveAfterDays in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the ApproveAfterDays is used to determine if the patch is approved for deployment.</td>
</tr>
</tbody>
</table>
Note
For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

3. The product of the instance is determined by SSM Agent. This attribute corresponds to the value of the Product key attribute in the patch baseline's PatchFilter data type.

4. Packages are selected for the update according to the following guidelines:

<table>
<thead>
<tr>
<th>Security option</th>
<th>Patch selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-defined default patch baselines provided by AWS and custom patch baselines where the <strong>Approved patches include non-security updates</strong> is <strong>not</strong> selected</td>
<td>For each update notice in updateinfo.xml, the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used. The equivalent yum command for this workflow is: <code>sudo yum update-minimal --sec-severity=important,moderate --bugfix -y</code></td>
</tr>
<tr>
<td>Custom patch baselines where the <strong>Approved patches include non-security updates</strong> is selected</td>
<td>In addition to applying the security updates that have been selected from updateinfo.xml, Patch Manager will apply nonsecurity updates that otherwise meet the patch filtering rules. The equivalent yum command for this workflow is: <code>sudo yum update --security --bugfix</code></td>
</tr>
</tbody>
</table>

For information about patch compliance status values, see About patch compliance status values (p. 946).

**How patch baseline rules work on RHEL**

On Red Hat Enterprise Linux, the patch selection process is as follows:

1. On the instance, the YUM library (RHEL 7) or the DNF library (RHEL 8) accesses the updateinfo.xml file for each configured repo.

   **Note**
   The updateinfo.xml file might not be available if the repo is not one managed by Red Hat. If there is no updateinfo.xml found, no patch will be applied.

2. Each update notice in updateinfo.xml includes several attributes that denote the properties of the packages in the notice, as described in the following table.
## Update notice attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| **type**  | Corresponds to the value of the Classification key attribute in the patch baseline's `PatchFilter` data type. Denotes the type of package included in the update notice.  
You can view the list of supported values by using the AWS CLI command `describe-patch-properties` or the API action `DescribePatchProperties`. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console. |
| **severity** | Corresponds to the value of the Severity key attribute in the patch baseline's `PatchFilter` data type. Denotes the severity of the packages included in the update notice. Usually only applicable for Security update notices.  
You can view the list of supported values by using the AWS CLI command `describe-patch-properties` or the API action `DescribePatchProperties`. You can also view the list in the Approval rules area of the Create patch baseline page or Edit patch baseline page in the Systems Manager console. |
| **update_id** | Denotes the advisory ID, such as `RHSA-2017:0864`. The advisory ID can be used in the `ApprovedPatches` or `RejectedPatches` attribute in the patch baseline. |
| **references** | Contains additional information about the update notice, such as a CVE ID (format: `CVE-2017-1000371`) or a Bugzilla ID (format: `1463241`). The CVE ID and Bugzilla ID can be used in the `ApprovedPatches` or `RejectedPatches` attribute in the patch baseline. |
| **updated** | Corresponds to `ApproveAfterDays` in the patch baseline. Denotes the released date (updated date) of the packages included in the update notice. A comparison between the current timestamp and the value of this attribute plus the `ApproveAfterDays` is used to determine if the patch is approved for deployment. |

**Note**
For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

3. The product of the instance is determined by SSM Agent. This attribute corresponds to the value of the Product key attribute in the patch baseline's `PatchFilter` data type.
4. Packages are selected for the update according to the following guidelines:

<table>
<thead>
<tr>
<th>Security option</th>
<th>Patch selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-defined default patch baselines provided by AWS and custom patch baselines</td>
<td>For each update notice in updateinfo.xml, the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used.</td>
</tr>
<tr>
<td>where the Approved patches include non-security updates is not selected</td>
<td>For RHEL 7, the equivalent yum command for this workflow is:</td>
</tr>
<tr>
<td></td>
<td>sudo yum update-minimal --sec-severity=critical,important --bugfix -y</td>
</tr>
<tr>
<td></td>
<td>For RHEL 8, the equivalent yum commands for this workflow is:</td>
</tr>
<tr>
<td></td>
<td>sudo dnf update-minimal --sec-severity=Critical --bugfix -y \ sudo dnf update-minimal --sec-severity=Important --bugfix -y</td>
</tr>
<tr>
<td>Custom patch baselines where the Approved patches include non-security updates is selected</td>
<td>In addition to applying the security updates that have been selected from updateinfo.xml, Patch Manager will apply nonsecurity updates that otherwise meet the patch filtering rules.</td>
</tr>
<tr>
<td></td>
<td>For RHEL 7, the equivalent yum command for this workflow is:</td>
</tr>
<tr>
<td></td>
<td>sudo yum update --security --bugfix</td>
</tr>
<tr>
<td></td>
<td>For RHEL 8, the equivalent dnf command for this workflow is:</td>
</tr>
<tr>
<td></td>
<td>sudo dnf update --security --bugfix</td>
</tr>
</tbody>
</table>

For information about patch compliance status values, see About patch compliance status values (p. 946).

How patch baseline rules work on SUSE Linux Enterprise Server

On SLES, each patch includes the following attributes that denote the properties of the packages in the patch:

- **Category**: Corresponds to the value of the Classification key attribute in the patch baseline's PatchFilter data type. Denotes the type of patch included in the update notice.

You can view the list of supported values by using the AWS CLI command `describe-patch-properties` or the API action `DescribePatchProperties`. You can also view the list in the Approval rules area of the **Create patch baseline** page or **Edit patch baseline** page in the Systems Manager console.
• **Severity**: Corresponds to the value of the **Severity** key attribute patch baseline's **PatchFilter** data type. Denotes the severity of the patches.

You can view the list of supported values by using the AWS CLI command **describe-patch-properties** or the API action **DescribePatchProperties**. You can also view the list in the **Approval rules** area of the **Create patch baseline** page or **Edit patch baseline** page in the Systems Manager console.

The product of the instance is determined by SSM Agent. This attribute corresponds to the value of the **Product** key attribute in the patch baseline's **PatchFilter** data type.

For each patch, the patch baseline is used as a filter, allowing only the qualified packages to be included in the update. If multiple packages are applicable after applying the patch baseline definition, the latest version is used.

**Note**
For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

**How patch baseline rules work on Ubuntu Server**

On Ubuntu Server, the patch baseline service offers filtering on the **Priority** and **Section** fields. These fields are typically present for all Ubuntu Server packages. To determine whether a patch is selected by the patch baseline, Patch Manager does the following:

1. On Ubuntu Server systems, the equivalent of `sudo apt-get update` is run to refresh the list of available packages. Repos are not configured and the data is pulled from repos configured in a `sources` list.
2. Next, the **GlobalFilters**, **ApprovalRules**, **ApprovedPatches** and **RejectedPatches** lists are applied.

**Note**
Because it's not possible to reliably determine the release dates of update packages for Ubuntu Server, the auto-approval options are not supported for this operating system.

Only packages with candidate versions appearing in the distribution security repo (archive) are selected. For Ubuntu Server 14 this is repo is `trusty-security`. For Ubuntu Server 16, it is `xenial-security`.

**Note**
For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

To view the contents of the **Priority** and **Section** fields, run the following **aptitude** command:

**Note**
You may need to first install Aptitude on Ubuntu Server 16 systems.

```bash
aptitude search -F '%p %P %s %t %V#' '~U'
```

In the response to this command, all upgradable packages are reported in this format:

| name, priority, section, archive, candidate version |

For information about patch compliance status values, see About patch compliance status values (p. 946).

**Key differences between Linux and Windows patching**

This topic describes important differences between Linux and Windows patching.
Note
To patch Linux instances, your instances must be running SSM Agent version 2.0.834.0 or later. An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.

Difference 1: Patch evaluation

Linux

For Linux patching, Systems Manager evaluates patch baseline rules and the list of approved and rejected patches on each managed instance. Systems Manager must evaluate patching on each instance because the service retrieves the list of known patches and updates from the repositories that are configured on the instance.

Windows

Patch Manager uses different processes on Windows managed instances and Linux managed instances in order to evaluate which patches should be present. For Windows patching, Systems Manager evaluates patch baseline rules and the list of approved and rejected patches directly in the service. It can do this because Windows patches are pulled from a single repository (Windows Update).

Difference 2: Not Applicable patches

Due to the large number of available packages for Linux operating systems, Systems Manager does not report details about patches in the Not Applicable state. A Not Applicable patch is, for example, a patch for Apache software when the instance does not have Apache installed. Systems Manager does report the number of Not Applicable patches in the summary, but if you call the DescribeInstancePatches API for an instance, the returned data does not include patches with a state of Not Applicable. This behavior is different from Windows.

Difference 3: SSM document support

The AWS-ApplyPatchBaseline SSM document doesn't support Linux instances. For applying patch baselines to both Windows Server and Linux instances, the recommended SSM document is AWS-RunPatchBaseline. For more information, see About SSM documents for patching instances (p. 935) and About the SSM document AWS-RunPatchBaseline (p. 938).

Difference 4: Application patches

Patch Manager's primary focus is applying patches to operating systems. However, you can also use Patch Manager to apply patches to some applications on your instances.

Linux

On Linux operating systems, Patch Manager uses the configured repositories for updates, and does not differentiate between operating systems and application patches. You can use Patch Manager to define which repositories to fetch updates from. For more information, see How to specify an alternative patch source repository (Linux) (p. 917).

Windows

On Windows Server instances, you can apply approval rules, as well as Approved and Rejected patch exceptions, for applications released by Microsoft, such as Microsoft Word 2011 and Microsoft Exchange Server 2016. For more information, see Create a custom patch baseline (p. 959).
About patching operations

The topics in this section provide information to help you understand how the Patch Manager service works.

Topics

- About patching configurations (p. 935)
- About SSM documents for patching instances (p. 935)
- About patch compliance status values (p. 946)

About patching configurations

A patching configuration defines a unique patching operation. The configuration specifies the instances for patching, which patch baseline is to be applied, the schedule for patching, and typically, the maintenance window that the configuration is to be associated with.

To create a patching configuration, use the Configure patching page. This page lets you associate a patching configuration with an existing maintenance window, create a new maintenance window for the configuration, or run a one-time manual patching operation on a set of instances. For more information, see Create a patching configuration (console) (p. 971).

About SSM documents for patching instances

This topic describes the seven SSM documents currently available to help you keep your managed instances patched with the latest security-related updates.

We currently recommend using just three of these documents in your patching operations. Together, these three SSM documents provide you with a full range of patching options using AWS Systems Manager. Two of these documents were released later than the four legacy SSM documents they replace and represent expansions or consolidations of functionality.

The three recommended SSM documents include:

- AWS-ConfigureWindowsUpdate
- AWS-InstallWindowsUpdates
- AWS-RunPatchBaseline

The four legacy SSM documents that are still available for use in some AWS Regions, but might be deprecated in the future, include:

- AWS-ApplyPatchBaseline
- AWS-FindWindowsUpdates
- AWS-InstallMissingWindowsUpdates
- AWS-InstallSpecificWindowsUpdates

Refer to the following sections for more information about using these SSM documents in your patching operations.

Topics

- SSM documents recommended for patching instances (p. 936)
- Legacy SSM documents for patching instances (p. 937)
- About the SSM document AWS-RunPatchBaseline (p. 938)
- Sample scenario for using the InstallOverrideList parameter in AWS-RunPatchBaseline (p. 946)
SSM documents recommended for patching instances

The following three SSM documents are recommended for use in your managed instance patching operations.

**Recommended SSM Documents**
- AWS-ConfigureWindowsUpdate (p. 936)
- AWS-InstallWindowsUpdates (p. 936)
- AWS-RunPatchBaseline (p. 936)

**AWS-ConfigureWindowsUpdate**

Supports configuring basic Windows Update functions and using them to install updates automatically (or to disable automatic updates). Available in all AWS Regions.

This SSM document prompts Windows Update to download and install the specified updates and reboot instances as needed. Use this document with State Manager to ensure Windows Update maintains its configuration. You can also run it manually using Run Command to change the Windows Update configuration.

The available parameters in this document support specifying a category of updates to install (or whether to disable automatic updates), as well as specifying the day of the week and time of day to run patching operations. This SSM document is most useful if you don't need strict control over Windows updates and don't need to collect compliance information.

**Replaces legacy SSM documents:**
- None

**AWS-InstallWindowsUpdates**

Installs updates on a Windows Server instance. Available in all AWS Regions.

This SSM document provides basic patching functionality in cases where you either want to install a specific update (using the `Include Kbs` parameter), or want to install patches with specific classifications or categories but don't need patch compliance information.

**Replaces legacy SSM documents:**
- AWS-FindWindowsUpdates
- AWS-InstallMissingWindowsUpdates
- AWS-InstallSpecificWindowsUpdates

The three legacy documents perform different functions, but you can achieve the same results by using different parameter settings with the newer SSM document AWS-InstallWindowsUpdates. These parameter settings are described in Legacy SSM documents for patching instances (p. 937).

**AWS-RunPatchBaseline**

Installs patches on your instances or scans instances to determine whether any qualified patches are missing. Available in all AWS Regions.

**AWS-RunPatchBaseline** enables you to control patch approvals using patch baselines. Reports patch compliance information that you can view using the Systems Manager Compliance tools. These tools provide you with insights on the patch compliance state of your instances, such as which instances are missing patches and what those patches are. For Linux operating systems, compliance information...
is provided for patches from both the default source repository configured on an instance and from any alternative source repositories you specify in a custom patch baseline. For more information about alternative source repositories, see How to specify an alternative patch source repository (Linux) (p. 917). For more information about the Systems Manager Compliance tools, see AWS Systems Manager Configuration Compliance (p. 696).

Replaces legacy documents:

- AWS-ApplyPatchBaseline

The legacy document AWS-ApplyPatchBaseline applies only to Windows Server instances, and does not provide support for application patching. The newer AWS-RunPatchBaseline provides the same support for both Windows and Linux systems. Version 2.0.834.0 or later of SSM Agent is required in order to use the AWS-RunPatchBaseline document.

For more information about the AWS-RunPatchBaseline SSM document, see About the SSM document AWS-RunPatchBaseline (p. 938).

Legacy SSM documents for patching instances

The following four SSM documents are still available for use in your patching operations in some AWS Regions. However, they might be deprecated in the future, so we do not recommend their use. Instead, use the documents described in SSM documents recommended for patching instances (p. 936).

Legacy SSM Documents

- AWS-ApplyPatchBaseline (p. 937)
- AWS-FindWindowsUpdates (p. 937)
- AWS-InstallMissingWindowsUpdates (p. 937)
- AWS-InstallSpecificWindowsUpdates (p. 938)

AWS-ApplyPatchBaseline

Supports only Windows Server instances, but does not include support for patching applications that is found in its replacement, AWS-RunPatchBaseline. Not available in AWS Regions launched after August 2017.

Note
The replacement for this SSM document, AWS-RunPatchBaseline, requires version 2.0.834.0 or a later version of SSM Agent. You can use the AWS-UpdateSSMAgent document to update your instances to the latest version of the agent.

AWS-FindWindowsUpdates

Replaced by AWS-InstallWindowsUpdates, which can perform all the same actions. Not available in AWS Regions launched after April 2017.

To achieve the same result that you would from this legacy SSM document, use the following parameter configuration with the recommended replacement document, AWS-InstallWindowsUpdates:

- Action = Scan
- Allow Reboot = False

AWS-InstallMissingWindowsUpdates

Replaced by AWS-InstallWindowsUpdates, which can perform all the same actions. Not available in any AWS Regions launched after April 2017.
To achieve the same result that you would from this legacy SSM document, use the following parameter configuration with the recommended replacement document, **AWS-InstallWindowsUpdates**:

- **Action** = Install
- **Allow Reboot** = True

**AWS-InstallSpecificWindowsUpdates**

Replaced by **AWS-InstallWindowsUpdates**, which can perform all the same actions. Not available in any AWS Regions launched after April 2017.

To achieve the same result that you would from this legacy SSM document, use the following parameter configuration with the recommended replacement document, **AWS-InstallWindowsUpdates**:

- **Action** = Install
- **Allow Reboot** = True
- **Include Kbs** = comma-separated list of KB articles

**About the SSM document AWS-RunPatchBaseline**

AWS Systems Manager supports an SSM document for Patch Manager, **AWS-RunPatchBaseline**, which performs patching operations on instances for both security related and other types of updates. You can use the document AWS-RunPatchBaseline to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for Microsoft applications.)

This document supports both Linux and Windows Server instances, so it can be reliably run on either type of instance when managed by Systems Manager. The document will perform the appropriate actions for each platform.

**Note**

Patch Manager also supports the legacy SSM document **AWS-ApplyPatchBaseline**. However, this document supports patching on Windows Server instances only. We encourage you to use **AWS-RunPatchBaseline** instead because it supports patching on both Linux and Windows Server instances. Version 2.0.834.0 or later of SSM Agent is required in order to use the **AWS-RunPatchBaseline** document.

**Windows**

On Windows Server instances, the **AWS-RunPatchBaseline** document downloads and invokes a PowerShell module, which in turn downloads a snapshot of the patch baseline that applies to the instance. This patch baseline snapshot is passed to the Windows Update API, which controls downloading and installing the approved patches as appropriate.

**Linux**

On Linux instances, the **AWS-RunPatchBaseline** document invokes a Python module, which in turn downloads a snapshot of the patch baseline that applies to the instance. This patch baseline snapshot uses the defined rules and lists of approved and blocked patches to drive the appropriate package manager for each instance type:

- Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and RHEL 7 instances use YUM. For YUM operations, Patch Manager requires Python 2.6 or later.
- RHEL 8 instances use DNF. For DNF operations, Patch Manager requires Python 2 or Python 3. (Neither version is installed by default on RHEL 8. You must install one or the other manually.)
- Debian and Ubuntu Server instances use APT. For APT operations, Patch Manager requires Python 3.
- SUSE Linux Enterprise Server instances use Zypper. For Zypper operations, Patch Manager requires Python 2.6 or later.
After all approved and applicable updates have been installed, with reboots performed as necessary, patch compliance information is generated on an instance and reported back to Patch Manager.

**Note**
If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: `RebootOption` (p. 945).

For information about viewing patch compliance data, see About patch compliance (p. 700).

**AWS-RunPatchBaseline parameters**

`AWS-RunPatchBaseline` supports four parameters. The `Operation` parameter is required. The `InstallOverrideList` and `RebootOption` parameters are optional. `Snapshot-ID` is technically optional, but we recommend that you supply a custom value for it when you run `AWS-RunPatchBaseline` outside of a maintenance window, and let Patch Manager supply the value automatically when the document is run as part of a maintenance window operation.

**Parameters**
- Parameter name: `Operation` (p. 939)
- Parameter name: `Snapshot ID` (p. 939)
- Parameter name: `InstallOverrideList` (p. 941)
- Parameter name: `RebootOption` (p. 945)

**Parameter name: `Operation`**

**Usage**: Required.

**Options**: Scan | Install.

**Scan**

When you choose the `Scan` option, `AWS-RunPatchBaseline` determines the patch compliance state of the instance and reports this information back to Patch Manager. `Scan` does not prompt updates to be installed or instances to be rebooted. Instead, the operation identifies where updates are missing that are approved and applicable to the instance.

**Install**

When you choose the `Install` option, `AWS-RunPatchBaseline` attempts to install the approved and applicable updates that are missing from the instance. Patch compliance information generated as part of an `Install` operation does not list any missing updates, but might report updates that are in a failed state if the installation of the update did not succeed for any reason. Whenever an update is installed on an instance, the instance is rebooted to ensure the update is both installed and active. (Exception: If the `RebootOption` parameter is set to `NoReboot` in the `AWS-RunPatchBaseline` document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: `RebootOption` (p. 945).)

**Note**
If a patch specified by the baseline rules is installed before Patch Manager updates the instance, the system might not reboot as expected. This can happen when a patch is installed manually by a user or installed automatically by another program, such as the `unattended-upgrades` package on Ubuntu Server.

**Parameter name: `Snapshot ID`**

**Usage**: Optional.
Snapshot ID is a unique ID (GUID) used by Patch Manager to ensure that a set of instances that are patched in a single operation all have the exact same set of approved patches. Although the parameter is defined as optional, our best practice recommendation depends on whether or not you are running `AWS-RunPatchBaseline` in a maintenance window, as described in the following table.

### AWS-RunPatchBaseline best practices

<table>
<thead>
<tr>
<th>Mode</th>
<th>Best practice</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running <code>AWS-RunPatchBaseline</code> inside a maintenance window</td>
<td>Do not supply a Snapshot ID. Patch Manager will supply it for you.</td>
<td>If you use a maintenance window to run <code>AWS-RunPatchBaseline</code>, you should not provide your own generated Snapshot ID. In this scenario, Systems Manager provides a GUID value based on the maintenance window execution ID. This ensures that a correct ID is used for all the invocations of <code>AWS-RunPatchBaseline</code> in that maintenance window. If you do specify a value in this scenario, note that the snapshot of the patch baseline might not remain in place for more than 24 hours. After that, a new snapshot will be generated even if you specify the same ID after the snapshot expires.</td>
</tr>
<tr>
<td>Running <code>AWS-RunPatchBaseline</code> outside of a maintenance window</td>
<td>Generate and specify a custom GUID value for the Snapshot ID.¹</td>
<td>When you are not using a maintenance window to run <code>AWS-RunPatchBaseline</code>, we recommend that you generate and specify a unique Snapshot ID for each patch baseline, particularly if you are running the <code>AWS-RunPatchBaseline</code> document on multiple instances in the same operation. If you do not specify an ID in this scenario, Systems Manager generates a different Snapshot ID for each instance the command is sent to. This might result in varying sets of patches being specified among the instances. For instance, say that you are running the <code>AWS-RunPatchBaseline</code> document directly via Run Command and targeting a group of 50 instances. Specifying a custom Snapshot ID results in the generation of a single baseline snapshot that is used</td>
</tr>
</tbody>
</table>

¹ If you are using the `AWS-RunPatchBaseline` document, you should specify a custom GUID value for the Snapshot ID. If you do not specify an ID, Systems Manager generates a different Snapshot ID for each instance the command is sent to. This might result in varying sets of patches being specified among the instances.
### About patching operations

<table>
<thead>
<tr>
<th>Mode</th>
<th>Best practice</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>to evaluate and patch all the instances, ensuring that they end up in a consistent state.</td>
</tr>
</tbody>
</table>

¹ You can use any tool capable of generating a GUID to generate a value for the Snapshot ID parameter. For example, in PowerShell, you can use the `New-Guid` cmdlet to generate a GUID in the format of `12345699-9405-4f69-bc5e-9315aEXAMPLE`.

**Parameter name: InstallOverrideList**

**Usage:** Optional.

`InstallOverrideList` lets you specify an HTTPS URL or an Amazon Simple Storage Service (Amazon S3) path-style URL to a list of patches to be installed. This patch installation list, which you maintain in YAML format, overrides the patches specified by the current default patch baseline. This provides you with more granular control over which patches are installed on your instances.

Be aware that compliance reports reflect patch states according to what's specified in the patch baseline, not what you specify in an `InstallOverrideList` list of patches. In other words, Scan operations ignore the `InstallOverrideList` parameter. This is to ensure that compliance reports consistently reflect patch states according to policy rather than what was approved for a specific patching operation.

For a description of how you might use the `InstallOverrideList` parameter to apply different types of patches to a target group, on different maintenance window schedules, while still using a single patch baseline, see Sample scenario for using the InstallOverrideList parameter in AWS-RunPatchBaseline (p. 946).

**Valid URL formats**

- **HTTPS URL format:**
  ```
  https://s3.amazonaws.com/my-patch-approval-lists-bucket/my-windows-override-list.yaml
  ```

- **Amazon S3 path-style URL:**
  ```
  s3://my-patch-approval-lists-bucket/my-windows-override-list.yaml
  ```

**Valid YAML content formats**

The formats you use to specify patches in your list depends on the operating system of your instance. The general format, however, is as follows:

```yaml
patches:
  - id: '{patch-d}'
    title: '{patch-title}'
    {additional-fields}:{values}
```

Although you can provide additional fields in your YAML file, they are ignored during patch operations.

In addition, we recommend verifying that the format of your YAML file is valid before adding or updating the list in your S3 bucket. For more information about the YAML format, see yaml.org. For validation tool options, perform a web search for "yaml format validators".
• **Linux**

  **id**

  The `id` field is required. Use it to specify patches using the package name and architecture. For example: `dhclient.x86_64`. You can use wildcards in `id` to indicate multiple packages. For example: `dhcP*` and `dhcP1.*`.

  **Title**

  The `title` field is optional, but on Linux systems it does provide additional filtering capabilities. If you use `title`, it should contain the package version information in the one of the following formats:

  **YUM/SUSE Linux Enterprise Server (SLES):**

  `{name}. {architecture}:{epoch}:{version}-{release}`

  **APT**

  `{name}. {architecture}:{version}`

  For Linux patch titles, you can use one or more wildcards in any position to expand the number of package matches. For example: `*32:9.8.2-0.*.rc1.57.amzn1`.

  For example:
  
  • **apt package version 1.2.25 is currently installed on your instance, but version 1.2.27 is now available.**
  
  • **You add apt.amd64 version 1.2.27 to the patch list. It depends on apt utils.amd64 version 1.2.27, but apt-utils.amd64 version 1.2.25 is specified in the list.**

  In this case, apt version 1.2.27 will be blocked from installation and reported as “Failed-NonCompliant.”

  • **Microsoft Windows**

    **id**

    The `id` field is required. Use it to specify patches using Microsoft Knowledge Base IDs (for example, KB2736693) and Microsoft Security Bulletin IDs (for example, MS17-023).

    Any other fields you want to provide in a patch list for Windows are optional and are for your own informational use only. You can use additional fields such as `title`, `classification`, `severity`, or anything else for providing more detailed information about the specified patches.

  **Other fields**

  Any other fields you want to provide in a patch list for Linux are optional and are for your own informational use only. You can use additional fields such as `classification`, `severity`, or anything else for providing more detailed information about the specified patches.

  **Sample patch lists**

  • **Amazon Linux**

    ```
    patches:
    - id: 'kernel.x86_64'
    ```

    942
About patching operations

- **CentOS**

  patches:
  - id: 'kernel.x86_64'
    title: '2.10.95-0ubuntu2.9'
  - id: 'cryptsetup-amd64'
    title: '*2:1.6.6-5ubuntu2.1'
  - id: 'cryptsetup-bin.*'
    title: '*2:1.6.6-5ubuntu2.1'
  - id: 'apt-amd64'
    title: '*1.2.27'
  - id: 'apt-utils-amd64'
    title: '*1.2.25'

- **Debian**

  patches:
  - id: 'apparmor-amd64'
    title: '2.10.95-0ubuntu2.9'
  - id: 'cryptsetup-amd64'
    title: '*2:1.6.6-5ubuntu2.1'
  - id: 'cryptsetup-bin.*'
    title: '*2:1.6.6-5ubuntu2.1'
  - id: 'apt-amd64'
    title: '*1.2.27'
  - id: 'apt-utils-amd64'
    title: '*1.2.25'

- **Oracle Linux**

  patches:
  - id: 'audit-libs.x86_64'
    title: '*2.8.5-4.el7'
  - id: 'curl.x86_64'
    title: '*el7'
  - id: 'grub2.x86_64'
    title: 'grub2.x86_64:1:2.02-0.81.0.1.el7'
  - id: 'grub2.x86_64'
    title: 'grub2.x86_64:1:*-0.81.0.1.el7'

- **Red Hat Enterprise Linux (RHEL)**
patches:
- id: 'NetworkManager.x86_64'
  title: '*1:1.10.2-14.el7_5'
- id: 'NetworkManager-*.*86_64'
  title: '*1:1.10.2-14.el7_5'
- id: 'audit.x86_64'
  title: '*0:2.8.1-3.el7'
- id: 'dhclient.x86_64'
  title: '*1:1.10.2-14.el7_5'
- id: 'dhcp*.x86_64'
  title: '*12:5.2.5-68.el7'

• SUSE Linux Enterprise Server (SLES)

patches:
- id: 'amazon-ssm-agent.x86_64'
- id: 'binutils'
  title: '*0:2.26.1-9.12.1'
- id: 'glibc*.x86_64'
  title: '*2.19*'
- id: 'dhcp*'
  title: '*1:1.10.2-14.el7_5'
- id: 'lib*'

• Ubuntu Server

patches:
- id: 'apparmor.amd64'
  title: '2.10.95-0ubuntu2.9'
- id: 'cryptsetup.amd64'
  title: '*2:1.6.6-5ubuntu2.1'
- id: 'cryptsetup-bin.*'
  title: '*2:1.6.6-5ubuntu2.1'
- id: 'apt.amd64'
  title: '*1.2.25'
- id: 'apt-utils.amd64'
  title: '*1.2.25'

• Windows

patches:
- id: 'KB4284819'
  title: '2018-06 Cumulative Update for Windows Server 2016 (1709) for x64-based Systems (KB4284819)'

944
Parameter name: RebootOption

Usage: Optional.

Options: RebootIfNeeded | NoReboot.

RebootIfNeeded

When you choose the RebootIfNeeded option, the instance is rebooted if Patch Manager installed new patches, or if it detected any patches with a status of INSTALLED_PENDING_REBOOT during the Install operation. The INSTALLED_PENDING_REBOOT status can mean that the option NoReboot was selected the last time the Install operation was run. (Patches installed outside of Patch Manager are never given a status of INSTALLED_PENDING_REBOOT.)

Note
When you choose the RebootIfNeeded option, Patch Manager does not evaluate whether a reboot is required by the patch. A reboot occurs whenever there are missing packages or packages with a status of INSTALLED_PENDING_REBOOT.

NoReboot

When you choose the NoReboot option, Patch Manager does not reboot an instance even if it installed patches during the Install operation. This option is useful if you know that your instances don’t require rebooting after patches are applied, or you have applications or processes running on an instance that should not be disrupted by a patching operation reboot. It is also useful when you want more control over the timing of instance reboots, such as by using a maintenance window.

Patch installation tracking file: To track patch installation, especially patches that have been installed since the last system reboot, Systems Manager maintains a file on the managed instance.

Important
Do not delete or modify the tracking file. If this file is deleted or corrupted, the patch compliance report for the instance is inaccurate. If this happens, reboot the instance and run a patch Scan operation to restore the file.

This tracking file is stored in the following locations on your managed instances:

- Linux operating systems: /var/log/amazon/ssm/patch-configuration/patch-states-configuration.json
- Windows Server operating system: C:\ProgramData\Amazon\PatchBaselineOperations\State\PatchStatesConfiguration.json
Sample scenario for using the InstallOverrideList parameter in AWS-RunPatchBaseline

You can use the InstallOverrideList parameter when you want to override the patches specified by the current default patch baseline. The following example shows how to use this parameter to achieve the following:

- Apply different sets of patches to a target group of instances.
- Apply these patch sets on different frequencies.
- Use the same patch baseline for both operations.

Say that you want to install two different categories of patches on your Amazon Linux 2 instances. You want to install these patches on different schedules using maintenance windows. You want one maintenance window to run every week and install all Security patches. You want another maintenance window to run once a month and install all available patches, or categories of patches other than Security.

However, only one patch baseline at a time can be defined as the default for an operating system. This requirement helps avoid situations where one patch baseline approves a patch while another blocks it, which can lead to issues between conflicting versions.

The following strategy lets you use the InstallOverrideList parameter to apply different types of patches to a target group, on different schedules, while still using the same patch baseline.

1. In the default patch baseline, ensure that only Security updates are specified.
2. Create a first maintenance window that runs AWS-RunPatchBaseline each week. Do not specify an override list.
3. Create an override list of the patches of all types that you want to apply on a monthly basis and store it in an Amazon S3 bucket. A sample script that you can modify to help create the list of patches follows this procedure.
4. Create a second maintenance window that runs once a month. But for the Run Command task you register for this maintenance window, specify the location of your override list.

The result: Only Security patches, as defined in your default patch baseline, are installed each week. All available patches, or whatever subset of patches you define, are installed each month.

Sample script for generating an override list

Download the following InstallOverrideList script and use it to generate an override list composed of all Missing patches reported from the most recent Scan operation. You can let the script generate a list of all patches, or you can use its filter fields to create a subset list. For example, you could specify that only patches of a certain classification or severity be added to the list. Note that the script uses the DescribeInstanceInformation command to create the list.

Download InstallOverrideList sample script

About patch compliance status values

After you use Systems Manager Patch Manager to install patches on your instances, compliance status information is immediately available to you in the console or in response to AWS CLI commands or corresponding Systems Manager API actions.

Note

If you want to assign a specific patch compliance status to an instance, you can use the put-compliance-items CLI command or the PutComplianceItems API action. Assigning compliance status is not supported in the console.
In the console, you can view patch compliance data in the following Systems Manager capabilities:

- Explorer (p. 118)
- Compliance (p. 696)
- Managed Instances (p. 758)

Using the AWS CLI, you can view summary information about patches on an instance by running commands such as the following:

- describe-instance-patch-states
- describe-instance-patch-states-for-patch-group
- describe-patch-group-state

**Patch compliance values for Debian and Ubuntu Server**

For Debian and Ubuntu Server, the rules for package classification into the different compliance states are as follows:

- **Installed**: Packages that are filtered through the patch baseline, with the candidate version appearing in trusty-security (Ubuntu Server 14) or xenial-security (Ubuntu Server 16), and are not upgradable.

- **Missing**: Packages that are filtered through the baseline, with the candidate version appearing in trusty-security (Ubuntu Server 14) or xenial-security (Ubuntu Server 16), and are upgradable.

- **Installed Other**: Packages that are not filtered through the baseline, with the candidate version appearing in trusty-security (Ubuntu Server 14) or xenial-security (Ubuntu Server 16), and are not upgradable. The compliance level for these packages is set to UNSPECIFIED.

- **NotApplicable**: Packages that are included in ApprovedPatches but are not installed on the system.

- **Failed**: Packages that failed to install during the patch operation.

**Patch compliance values for other operating systems**

For all operating systems besides Debian and Ubuntu Server, the system reports one of the following compliance status values for each patch:

- **INSTALLED**: The patch is listed in the patch baseline and is installed on the instance. It could have been installed either manually by an individual or automatically by Patch Manager when the AWS-RunPatchBaseline document was run on the instance.

- **INSTALLED_OTHER**: The patch is not in the baseline, but it is installed on the instance. An individual might have installed it manually.

- **INSTALLED_REJECTED**: The patch is installed on the instance but is specified in a rejected patches list. This typically means the patch was installed before it was added to a list of rejected patches.

- **INSTALLED_PENDING_REBOOT**: The Patch Manager Install operation applied the patch to the instance (or a patch was applied to a Windows Server instance outside of Patch Manager), but the instance has not been rebooted since the patch was applied. (Note that patches installed outside of Patch Manager are never given a status of INSTALLED_PENDING_REBOOT.) This typically means the NoReboot option was selected for the RebootOption parameter when the AWS-RunPatchBaseline document was last run on the instance. For more information, see Parameter name: RebootOption (p. 945).

- **MISSING**: The patch is approved in the baseline, but it's not installed on the instance. If you configure the AWS-RunPatchBaseline document task to scan (instead of install), the system reports this status for patches that were located during the scan but have not been installed.
About patch baselines

The topics in this section provide information about how patch baselines work.

**Topics**

- About predefined and custom patch baselines (p. 948)
- About package name formats for approved and rejected patch lists (p. 952)
- About patch groups (p. 953)
- About patching schedules using Maintenance Windows (p. 956)
- About patching applications on Windows Server (p. 957)

About predefined and custom patch baselines

A patch baseline defines which patches are approved for installation on your instances. You can specify approved or rejected patches one by one. You can also create auto-approval rules to specify that certain types of updates (for example, critical updates) should be automatically approved. The rejected list overrides both the rules and the approve list.

To use a list of approved patches to install specific packages, you first remove all auto-approval rules. If you explicitly identify a patch as rejected, it will not be approved or installed, even if it matches all of the criteria in an auto-approval rule. Also, a patch is installed on an instance only if it applies to the software on the instance, even if the patch has otherwise been approved for the instance.

Patch Manager provides predefined patch baselines for each of the operating systems supported by Patch Manager. You can use these baselines as they are currently configured (you can't customize them) or you can create your own patch baselines if you want greater control over which patches are approved or rejected for your environment.

**Topics**

- About predefined baselines (p. 948)
- About custom baselines (p. 950)

About predefined baselines

The following table describes the predefined patch baselines provided with Patch Manager.

For information about which versions of each operating system Patch Manager supports, see Patch Manager prerequisites (p. 912).
<table>
<thead>
<tr>
<th>Name</th>
<th>Supported operating system</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS-AmazonLinuxDefaultPatchBaseline</td>
<td>Amazon Linux</td>
<td>Approves all operating system patches that are classified as &quot;Security&quot; and that have a severity level of &quot;Critical&quot; or &quot;Important&quot;. Patches are auto-approved seven days after release. Also auto-approves all patches with a classification of &quot;Bugfix&quot; seven days after release.</td>
</tr>
<tr>
<td>AWS-AmazonLinux2DefaultPatchBaseline</td>
<td>Amazon Linux 2</td>
<td>Approves all operating system patches that are classified as &quot;Security&quot; and that have a severity level of &quot;Critical&quot; or &quot;Important&quot;. Patches are auto-approved seven days after release. Also approves all patches with a classification of &quot;Bugfix&quot; seven days after release.</td>
</tr>
<tr>
<td>AWS-CentOSDefaultPatchBaseline</td>
<td>CentOS</td>
<td>Approves all updates seven days after they become available, including nonsecurity updates.</td>
</tr>
<tr>
<td>AWS-DebianDefaultPatchBaseline</td>
<td>Debian</td>
<td>Immediately approves all operating system security-related patches that have a priority of &quot;Required&quot;, &quot;Important&quot;, &quot;Standard,&quot; &quot;Optional,&quot; or &quot;Extra.&quot; There is no wait before approval because reliable release dates are not available in the repos.</td>
</tr>
<tr>
<td>AWS-OracleLinuxDefaultPatchBaseline</td>
<td>Oracle Linux</td>
<td>Approves all operating system patches that are classified as &quot;Security&quot; and that have a severity level of &quot;Important&quot; or &quot;Moderate&quot;. Patches are auto-approved seven days after release. Also approves all patches that are classified as &quot;Bugfix&quot; seven days after release.</td>
</tr>
<tr>
<td>AWS-RedHatDefaultPatchBaseline</td>
<td>Red Hat Enterprise Linux (RHEL)</td>
<td>Approves all operating system patches that are classified as &quot;Security&quot; and that have a severity level of &quot;Critical&quot; or &quot;Important&quot;. Patches are auto-approved seven days after release. Also approves all patches that are classified as &quot;Bugfix&quot; seven days after release.</td>
</tr>
</tbody>
</table>
| AWS-SuseDefaultPatchBaseline   | SUSE Linux Enterprise Server (SLES) | Approves all operating system patches that are classified as }
# About patch baselines

<table>
<thead>
<tr>
<th>Name</th>
<th>Supported operating system</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS-UbuntuDefaultPatchBaseline</td>
<td>Ubuntu Server</td>
<td>“Security” and with a severity of &quot;Critical&quot; or &quot;Important&quot;. Patches are auto-approved seven days after release.</td>
</tr>
<tr>
<td>AWS-DefaultPatchBaseline</td>
<td>Windows Server</td>
<td>Immediately approves all operating system security-related patches that have a priority of &quot;Required&quot;, &quot;Important&quot;, &quot;Standard,&quot; &quot;Optional,&quot; or &quot;Extra.&quot; There is no wait before approval because reliable release dates are not available in the repos.</td>
</tr>
<tr>
<td>AWS-DefaultPatchBaseline-OS</td>
<td>Windows Server</td>
<td>Approves all Windows Server operating system patches that are classified as &quot;CriticalUpdates&quot; or &quot;SecurityUpdates&quot; and that have an MSRC severity of &quot;Critical&quot; or &quot;Important&quot;. Patches are auto-approved seven days after release.</td>
</tr>
<tr>
<td>AWS-DefaultPatchBaseline-OS-Applications</td>
<td>Windows Server</td>
<td>Approves all Windows Server operating system patches that are classified as &quot;CriticalUpdates&quot; or &quot;SecurityUpdates&quot; and that have an MSRC severity of &quot;Critical&quot; or &quot;Important&quot;. Patches are auto-approved seven days after release.</td>
</tr>
<tr>
<td>AWS-PredefinedPatchBaseline-OS</td>
<td>Windows Server</td>
<td>For the Windows Server operating system, approves all patches that are classified as &quot;CriticalUpdates&quot; or &quot;SecurityUpdates&quot; and that have an MSRC severity of &quot;Critical&quot; or &quot;Important&quot;. For Microsoft applications, approves all patches. Patches for both OS and applications are auto-approved seven days after release.</td>
</tr>
</tbody>
</table>

### About custom baselines

If you create your own patch baseline, you can choose which patches to auto-approve by using the following categories.

- Operating system: Windows, Amazon Linux, Ubuntu Server, and so on.
- Product name (for operating systems): For example, RHEL 6.5, Amazon Linux 2014.09, Windows Server 2012, Windows Server 2012 R2, and so on.
• Product name (for Microsoft applications on Windows Server only): For example, Word 2016, BizTalk Server, and so on.
• Classification: For example, critical updates, security updates, and so on.
• Severity: For example, critical, important, and so on.

For each approval rule that you create, you can choose to specify an auto-approval delay or specify a patch approval cutoff date.

**Note**
Because it's not possible to reliably determine the release dates of update packages for Ubuntu Server, the auto-approval options are not supported for this operating system.

An auto-approval delay is the number of days to wait after the patch was released, before the patch is automatically approved for patching. For example, if you create a rule using the CriticalUpdates classification and configure it for seven days auto-approval delay, then a new critical patch released on July 7 is automatically approved on July 14.

**Note**
If a Linux repository doesn't provide release date information for packages, Systems Manager uses the build time of the package as the auto-approval delay for Amazon Linux, Amazon Linux 2, RHEL, and CentOS. If the system isn't able to find the build time of the package, Systems Manager treats the auto-approval delay as having a value of zero.

When you specify an auto-approval cutoff date, Patch Manager automatically applies all patches released on or before that date. For example, if you specify July 7, 2020, as the cutoff date, no patches released on or after July 8, 2020, are installed automatically.

You can also specify a compliance severity level. If an approved patch is reported as missing, Compliance Level is the severity of the compliance violation.

By using multiple patch baselines with different auto-approval delays or cutoff dates, you can deploy patches at different rates to different instances. For example, you can create separate patch baselines, auto-approval delays, and cutoff dates for development and production environments. This enables you to test patches in your development environment before they get deployed in your production environment.

Keep the following in mind when you create a patch baseline:

• Patch Manager provides one predefined patch baseline for each supported operating system. These predefined patch baselines are used as the default patch baselines for each operating system type unless you create your own patch baseline and designate it as the default for the corresponding operating system type.

**Note**
For Windows Server, two predefined patch baselines are provided. The patch baseline AWS–WindowsPredefinedPatchBaseline–OS supports only operating system updates on the Windows operating system itself. It is used as the default patch baseline for Windows Server instances unless you specify a different patch baseline. The other predefined Windows patch baseline, AWS–WindowsPredefinedPatchBaseline–OS–Applications, can be used to apply patches to both the Windows Server operating system and supported Microsoft applications.

• For on-premises or other non-EC2 instances, Patch Manager attempts to use your custom default patch baseline. If no custom default patch baseline exists, the system uses the predefined patch baseline for the corresponding operating system.

• If a patch is listed as both approved and rejected in the same patch baseline, the patch is rejected.

• An instance can have only one patch baseline defined for it.

• The formats of package names you can add to lists of approved patches and rejected patches for a patch baseline depend on the type of operating system you are patching.
For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

For information about creating a patch baseline, see Create a custom patch baseline (p. 959) and Walkthrough: Patch a server environment (command line) (p. 991).

**About package name formats for approved and rejected patch lists**

The formats of package names you can add to lists of approved patches and rejected patches depend on the type of operating system you are patching.

**Package name formats for Linux operating systems**

The formats you can specify for approved and rejected patches in your patch baseline vary by Linux type. More specifically, the formats that are supported depend on the package manager used by the type of Linux operating system.

**Topics**

- Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and Red Hat Enterprise Linux (RHEL) (p. 952)
- Debian and Ubuntu Server (p. 953)
- SUSE Linux Enterprise Server (SLES) (p. 953)

**Amazon Linux, Amazon Linux 2, CentOS, Oracle Linux, and Red Hat Enterprise Linux (RHEL)**

**Package manager:** YUM, except for RHEL 8, which uses DNF as the package manager

**Approved patches:** For approved patches, you can specify any of the following:

- Bugzilla IDs, in the format 1234567 (The system processes numbers-only strings as Bugzilla IDs.)
- CVE IDs, in the format CVE-2018-1234567
- Advisory IDs, in formats such as RHSA-2017:0864 and ALAS-2018-123
- Full package names, in formats such as:
  - example-pkg-0.710.10-2.7.abcd.x86_64
  - pkg-example-EE-20180914-2.2.amzn1.noarch
- Package-names with a single wildcard, in formats such as:
  - example-pkg-*.abcd.x86_64
  - example-pkg-*-20180914-2.2.amzn1.noarch
  - example-pkg-EE-2018*.amzn1.noarch

**Rejected patches:** For rejected patches, you can specify any of the following:

- Full package names, in formats such as:
  - example-pkg-0.710.10-2.7.abcd.x86_64
  - pkg-example-EE-20180914-2.2.amzn1.noarch
- Package-names with a single wildcard, in formats such as:
  - example-pkg-*.abcd.x86_64
  - example-pkg-*-20180914-2.2.amzn1.noarch
  - example-pkg-EE-2018*.amzn1.noarch
• example-pkg-EE-2018*.amzn1.noarch

**Debian and Ubuntu Server**

**Package manager:** APT

**Approved patches and rejected patches:** For both approved and rejected patches, specify the following:

- Package names, in the format `ExamplePkg33`

  **Note**
  For Ubuntu Server lists, do not include elements such as architecture or versions. For example, you specify the package name `ExamplePkg33` to include all the following in a patch list:
  - `ExamplePkg33.x86.1`
  - `ExamplePkg33.x86.2`
  - `ExamplePkg33.x64.1`
  - `ExamplePkg33.3.2.5-364.noarch`

**SUSE Linux Enterprise Server (SLES)**

**Package manager:** Zypper

**Approved patches and rejected patches:** For both approved and rejected patch lists, you can specify any of the following:

- Full package names, in formats such as:
  - `SUSE-SLE-Example-Package-12-2018-123`
  - `example-pkg-2018.11.4-46.17.1.x86_64.rpm`

- Package names with a single wildcard, such as:
  - `SUSE-SLE-Example-Package-12-2018-*`
  - `example-pkg-2018.11.4-46.17.1.*.rpm`

**Package name formats for Windows operating systems**

For Windows operating systems, specify patches using Microsoft Knowledge Base IDs and Microsoft Security Bulletin IDs; for example:

```
KB2032276,KB2124261,MS10-048
```

**About patch groups**

You can use a *patch group* to associate instances with a specific patch baseline. Patch groups help ensure that you are deploying the appropriate patches, based on the associated patch baseline rules, to the correct set of instances. Patch groups can also help you avoid deploying patches before they have been adequately tested. For example, you can create patch groups for different environments (such as Development, Test, and Production) and register each patch group to an appropriate patch baseline.

**Note**
A patch group can only be registered with one patch baseline.

When you run `AWS-RunPatchBaseline`, you can target managed instances using their instance ID or tags. SSM Agent and Patch Manager then evaluate which patch baseline to use based on the patch group value that you added to the instance.
You create a patch group by using Amazon EC2 tags. Unlike other tagging scenarios across Systems Manager, a patch group must be defined with the tag key: **Patch Group**. Note that the key is case-sensitive. You can specify any value, for example "web servers," but the key must be **Patch Group**.

**Note**
An instance can only be in one patch group.

After you create a patch group and tag instances, you can register the patch group with a patch baseline. Registering the patch group with a patch baseline ensures that the instances within the patch group use the rules defined in the associated patch baseline. For more information on how to create a patch group and associate the patch group to a patch baseline, see [Create a patch group](p. 965) and [Add a patch group to a patch baseline](p. 967).

To view an example of creating a patch baseline and patch groups by using the AWS CLI, see [Walkthrough: Patch a server environment (command line)](p. 991). For more information about Amazon EC2 tags, see [Tagging your Amazon EC2 resources](p. 991) in the *Amazon EC2 User Guide*.

**How it works**

When the system runs the task to apply a patch baseline to an instance, SSM Agent verifies that a patch group value is defined for the instance. If the instance is assigned to a patch group, Patch Manager then verifies which patch baseline is registered to that group. If a patch baseline is found for that group, Patch Manager notifies SSM Agent to use the associated patch baseline. If an instance isn't configured for a patch group, Patch Manager automatically notifies SSM Agent to use the currently configured default patch baseline.

The following diagram shows a general example of the processes that Systems Manager performs when sending a Run Command task to your fleet of servers to patch using Patch Manager. A similar process is used when a maintenance window is configured to send a command to patch using Patch Manager.

In this example, we have three groups of EC2 instances for Windows Server with the following tags applied:

<table>
<thead>
<tr>
<th>EC2 instances group</th>
<th>Tags</th>
</tr>
</thead>
</table>
| Group 1             | key=OS,value=Windows  
                     | key=Patch Group,value=DEV |
| Group 2             | key=OS,value=Windows  |
| Group 3             | key=OS,value=Windows  
                     | key=Patch Group,value=QA |

For this example, we also have these two Windows patch baselines:

<table>
<thead>
<tr>
<th>Patch baseline ID</th>
<th>Default</th>
<th>Associated patch group</th>
</tr>
</thead>
<tbody>
<tr>
<td>pb-0123456789abcdef0</td>
<td>Yes</td>
<td>Default</td>
</tr>
<tr>
<td>pb-9876543210abcdef0</td>
<td>No</td>
<td>DEV</td>
</tr>
</tbody>
</table>

*Diagram 1: General Example of Patching Operations Process Flow*
The general process to scan or install patches using Run Command and Patch Manager is as follows:

1. **Send a command to patch**: Use the Systems Manager console, SDK, AWS CLI, or AWS Tools for Windows PowerShell to send a Run Command task using the document `AWS-RunPatchBaseline`. The diagram shows a Run Command task to patch managed instances by targeting the tag `key=OS,value=Windows`.

2. **Patch baseline determination**: SSM Agent verifies the patch group tags applied to the EC2 instance and queries Patch Manager for the corresponding patch baseline.
   - **Matching patch group value associated with patch baseline**: 
     1. SSM Agent, which is installed on EC2 instances in group one, receives the command issued in Step 1 to begin a patching operation. SSM Agent validates that the EC2 instances have the patch group tag-value `DEV` applied and queries Patch Manager for an associated patch baseline.
2. Patch Manager verifies that patch baseline `pb-9876543210abcdef0` has the patch group `DEV` associated and notifies SSM Agent.

3. SSM Agent retrieves a patch baseline snapshot from Patch Manager based on the approval rules and exceptions configured in `pb-9876543210abcdef0` and proceeds to the next step.

**No patch group tag added to instance:**

1. SSM Agent, which is installed on EC2 instances in group two, receives the command issued in Step 1 to begin a patching operation. SSM Agent validates that the EC2 instances don't have a Patch Group tag applied and as a result, SSM Agent queries Patch Manager for the default Windows patch baseline.

2. Patch Manager verifies that the default Windows Server patch baseline is `pb-0123456789abcdef0` and notifies SSM Agent.

3. SSM Agent retrieves a patch baseline snapshot from Patch Manager based on the approval rules and exceptions configured in the default patch baseline `pb-0123456789abcdef0` and proceeds to the next step.

**No matching patch group value associated with a patch baseline:**

1. SSM Agent, which is installed on EC2 instances in group three, receives the command issued in Step 1 to begin a patching operation. SSM Agent validates that the EC2 instances have the patch group tag-value `QA` applied and queries Patch Manager for an associated patch baseline.

2. Patch Manager does not find a patch baseline that has the patch group `QA` associated.

3. Patch Manager notifies SSM Agent to use the default Windows patch baseline `pb-0123456789abcdef0`.

4. SSM Agent retrieves a patch baseline snapshot from Patch Manager based on the approval rules and exceptions configured in the default patch baseline `pb-0123456789abcdef0` and proceeds to the next step.

3. **Patch scan or install:** After determining the appropriate patch baseline to use, SSM Agent begins either scanning for or installing patches based on the operation value specified in Step 1. The patches that are scanned for or installed are determined by the approval rules and patch exceptions defined in the patch baseline snapshot provided by Patch Manager.

**Related content**

About patch compliance status values (p. 946)

**About patching schedules using Maintenance Windows**

After you configure a patch baseline (and optionally a patch group), you can apply patches to your instance by using a maintenance window. A maintenance window can reduce the impact on server availability by letting you specify a time to perform the patching process that doesn't interrupt business operations. A maintenance window works like this:

1. Create a maintenance window with a schedule for your patching operations.

2. Choose the targets for the maintenance window by specifying the Patch Group tag for the tag name, and any value for which you have defined Amazon EC2 tags, for example, "production servers".

3. Create a new maintenance window task, and specify the AWS-RunPatchBaseline document.

When you configure the task, you can choose to either scan instances or scan and install patches on the instances. If you choose to scan instances, Patch Manager scans each instance and generates a list of missing patches for you to review.

If you choose to scan and install patches, Patch Manager scans each instance and compares the list of installed patches against the list of approved patches in the baseline. Patch Manager identifies missing patches, and then downloads and installs all missing and approved patches.
If you want to perform a one-time scan or install to fix an issue, you can use Run Command to call the AWS-RunPatchBaseline document directly.

**Important**
After installing patches, Systems Manager reboots each instance. The reboot is required to make sure that patches are installed correctly and to ensure that the system did not leave the instance in a potentially bad state. (Exception: If the RebootOption parameter is set to NoReboot in the AWS-RunPatchBaseline document, the instance is not rebooted after Patch Manager runs. For more information, see Parameter name: RebootOption (p. 945).)

**About patching applications on Windows Server**

For Windows Server, two predefined patch baselines are provided. The patch baseline AWS-WindowsPredefinedPatchBaseline-OS supports only operating system updates on the Windows operating system itself. It is used as the default patch baseline for Windows Server instances unless you specify a different patch baseline. The other predefined Windows patch baseline, AWS-WindowsPredefinedPatchBaseline-OS-Applications, can be used to apply patches to both the Windows Server operating system and supported Microsoft applications.

You can also create a custom patch baseline to update Microsoft applications on Windows Server machines.

To include Microsoft applications in your custom patch baseline, you must, at a minimum, specify the product that you want to patch. The following AWS CLI command demonstrates the minimal requirements to patch a product, such as Office 2016:

**Linux**

```
aws ssm create-patch-baseline \
   --name "My-Windows-App-Baseline" \
   --approval-rules
   "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=PRODUCT,Values='Office 2016'}, {Key=PATCH_SET,Values='APPLICATION'}]},ApproveAfterDays=5}]
```

**Windows**

```
aws ssm create-patch-baseline ^
   --name "My-Windows-App-Baseline" ^
   --approval-rules
   "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=PRODUCT,Values='Office 2016'}, {Key=PATCH_SET,Values='APPLICATION'}]},ApproveAfterDays=5}]
```

If you specify the Microsoft application product family, each product you specify must be a supported member of the selected product family. For example, to patch the product "Active Directory Rights Management Services Client 2.0," you must specify its product family as "Active Directory" and not, for example, "Office" or "SQL Server." The following AWS CLI command demonstrates a match pairing of product family and product:

**Linux**

```
aws ssm create-patch-baseline \
   --name "My-Windows-App-Baseline" \
   --approval-rules
   "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=PRODUCT_FAMILY,Values='Active Directory'},{Key=PRODUCT,Values='Active Directory Rights Management Services Client 2.0'},{Key=PATCH_SET,Values='APPLICATION'}]},ApproveAfterDays=5}]
```

957
Windows

```
aws ssm create-patch-baseline ^
  --name "My-Windows-App-Baseline" ^
  --approval-rules
  "PatchRules=[{PatchFilterGroup=,{PatchFilters=,[{Key=PRODUCT_FAMILY,Values='Active
  Directory'},{Key=PRODUCT,Values='Active Directory Rights Management Services Client
  2.0'},{Key=PATCH_SET,Values='APPLICATION'}],},ApproveAfterDays=5}]
```

Troubleshooting Mismatched Product Family/Product Pairs

When you create a patch baseline in the console, you specify a product family and a product. For
example, you might choose:

- **Product family**: Office
  - **Product**: Office 2016

If you attempt to create a patch baseline with a mismatched product family/product pair, an error
message is displayed. The following are reasons this can occur:

- You selected a valid product family/product pair, but then removed the product family selection.
- You chose a product from the **Obsolete or mismatched options** sublist instead of the **Available and matching options** sublist.

Items in the product **Obsolete or mismatched options** sublist might have been entered in error
through an SDK or AWS CLI `create-patch-baseline` command. This could mean a typo was
introduced or a product was assigned to the wrong product family. A product also appears in the
**Obsolete or mismatched options** sublist if it was specified for a previous patch baseline but currently
has no patches available from Microsoft.

To avoid this issue in the console, always choose options from the **Currently available options** sublists.

You can also view the products that have currently available patches by using the `describe-patch-
properties` command in the AWS CLI or the **DescribePatchProperties** API command.

Working with Patch Manager (console)

To use Patch Manager, complete the following tasks. These tasks are described in more detail in this
section.

1. Verify that the AWS predefined patch baseline for each operating system type that you use meets
   your needs. If it does not, create a patch baseline that defines a standard set of patches for that
   instance type and set it as the default instead.
2. Organize instances into patch groups by using Amazon EC2 tags (optional, but recommended).
3. Schedule patching by using a maintenance window that defines which instances to patch and when to
   patch them.
4. Monitor patching to verify compliance and investigate failures.

Related Content

- To view an example of how to create a patch baseline, patch groups, and a maintenance window using
  the AWS CLI, see **Walkthrough: Patch a server environment (command line)** (p. 991).
• For more information about maintenance windows, see AWS Systems Manager Maintenance Windows (p. 620).
• For information about monitoring patch compliance, see About patch compliance (p. 700).

Topics
• View AWS predefined patch baselines (p. 959)
• Create a custom patch baseline (p. 959)
• Set an existing patch baseline as the default (p. 965)
• Create a patch group (p. 965)
• Create a maintenance window for patching (p. 968)
• Create a patching configuration (console) (p. 971)
• Update or delete a patch baseline (console) (p. 972)

View AWS predefined patch baselines
Patch Manager includes a predefined patch baseline for each operating system supported by Patch Manager. You can use these patch baselines (you can't customize them), or you can create your own. The following procedure describes how to view a predefined patch baseline to see if it meets your needs. To learn more about patch baselines, see About predefined and custom patch baselines (p. 948).

To view AWS predefined patch baselines
2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Patch Manager.
3. In the patch baselines list, choose the baseline ID of one of the predefined patch baselines.
   Note
   For Windows Server, two predefined patch baselines are provided. The patch baseline AWS-WindowsPredefinedPatchBaseline-OS supports only operating system updates on the Windows operating system itself. It is used as the default patch baseline for Windows Server instances unless you specify a different patch baseline. The other predefined Windows patch baseline, AWS-WindowsPredefinedPatchBaseline-OS-Applications, can be used to apply patches to both the Windows Server operating system and supported Microsoft applications.
   For more information, see Set an existing patch baseline as the default (p. 965).
4. Choose the Approval rules tab and review the patch baseline configuration.
5. If the configuration is acceptable for your instances, you can skip ahead to the procedure Create a patch group (p. 965).
   -or-
   To create your own default patch baseline, continue to the topic Create a custom patch baseline (p. 959).

Create a custom patch baseline
Patch Manager includes a predefined patch baseline for each operating system supported by Patch Manager. You can use these patch baselines (you can't customize them), or you can create your own. The
following procedures describe how to create your own custom patch baseline. To learn more about patch baselines, see About predefined and custom patch baselines (p. 948).

Depending on the type of operating system you are using, Windows or Linux, use one of the following procedures.

Topics
- Create a custom patch baseline (Windows) (p. 960)
- Create a custom patch baseline (Linux) (p. 962)

Create a custom patch baseline (Windows)

Use the following procedure to create a custom patch baseline for Windows instances. For information about creating a patch baseline for Linux instances, see Create a custom patch baseline (Linux) (p. 962).

For an example of creating a patch baseline that is limited to installing Windows Service Packs only, see Walkthrough: Create a patch baseline for installing Windows Service Packs (console) (p. 990).

To create a custom patch baseline (Windows)

2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Patch Manager.
3. Choose Create patch baseline.
4. For Name, enter a name for your new patch baseline, for example, MyWindowsPatchBaseline.
5. (Optional) For Description, enter a description for this patch baseline.
6. For Operating system, choose Windows.
7. If you want to begin using this patch baseline as the default for Windows as soon as you create it, select Set this patch baseline as the default patch baseline for Windows Server instances.
   If you choose not to set this patch baseline for use now, you can do so later. For information, see Set an existing patch baseline as the default (p. 965).
8. In the Approval rules for operating systems section, use the fields to create one or more auto-approval rules.
   - Product: The version of the operating systems the approval rule applies to, such as WindowsServer2008. The default selection is All.
   - Classification: The type of patches the approval rule applies to, such as CriticalUpdates, Drivers, and Tools. The default selection is All.
   Tip
   You can include Windows Service Pack installations in your approval rules by including ServicePacks or by choosing All in your Classification list. For an example, see Walkthrough: Create a patch baseline for installing Windows Service Packs (console) (p. 990).
   - Severity: The severity value of patches the rule is to apply to, such as Critical. The default selection is All.
   - Auto-approval: The method for selecting patches for automatic approval.
   - Approve patches after a specified number of days: The number of days for Patch Manager to wait after a patch is released before a patch is automatically approved. You can enter any integer from zero (0) to 100.
• **Approve patches released up to a specific date**: The patch release date for which Patch Manager automatically applies all patches released on or before that date. For example, if you specify July 7, 2020, no patches released on or after July 8, 2020, are installed automatically.

• (Optional) **Compliance reporting**: The severity level you want to assign to patches approved by the baseline, such as High.

  **Note**  
  If an approved patch is reported as missing, the option you choose in Compliance reporting, such as Critical or Medium, determines the severity of the compliance violation.

9. In the Approval rules for applications section, use the fields to create one or more auto-approval rules.

• **Product family**: The general Microsoft product family for which you want to specify a rule, such as Office or Exchange Server.

• **Product**: The version of the application the approval rule applies to, such as Office 2016 or Active Directory Rights Management Services Client 2.0 2016. The default selection is All.

• **Classification**: The type of patches the approval rule applies to, such as CriticalUpdates. The default selection is All.

• **Severity**: The severity value of patches the rule applies to, such as Critical. The default selection is All.

• **Auto-approval**: The method for selecting patches for automatic approval.

  • **Approve patches after a specified number of days**: The number of days for Patch Manager to wait after a patch is released before a patch is automatically approved. You can enter any integer from zero (0) to 100.

  • **Approve patches released up to a specific date**: The patch release date for which Patch Manager automatically applies all patches released on or before that date. For example, if you specify July 7, 2020, no patches released on or after July 8, 2020, are installed automatically.

  • (Optional) **Compliance reporting**: The severity level you want to assign to patches approved by the baseline, such as High.

    **Note**  
    If an approved patch is reported as missing, the option you choose in Compliance reporting, such as Critical or Medium, determines the severity of the compliance violation.

10. If you want to explicitly approve any patches in addition to those meeting your approval rules, do the following in the Patch exceptions section:

• For Approved patches, enter a comma-separated list of the patches you want to approve.

  **Note**  
  For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).

• (Optional) For Approved patches compliance level, assign a compliance level to the patches in the list.

11. If you want to explicitly reject any patches that otherwise meet your approval rules, do the following in the Patch exceptions section:

• For Rejected patches, enter a comma-separated list of the patches you want to reject.

  **Note**  
  For information about accepted formats for lists of approved patches and rejected patches, see About package name formats for approved and rejected patch lists (p. 952).
For **Rejected patches action**, select the action for Patch Manager to take on patches included in the **Rejected patches** list.

- **Allow as dependency**: A package in the **Rejected patches** list is installed only if it is a dependency of another package. It is considered compliant with the patch baseline and its status is reported as **InstalledOther**. This is the default action if no option is specified.
- **Block**: Packages in the **Rejected patches** list, and packages that include them as dependencies, are not installed under any circumstances. If a package was installed before it was added to the **Rejected patches** list, it is considered noncompliant with the patch baseline and its status is reported as **InstalledRejected**.

12. (Optional) For **Manage tags**, apply one or more tag key name/value pairs to the patch baseline.

Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a patch baseline to identify the severity level of patches it specifies, the operating system family it applies to, and the environment type. In this case, you could specify tags similar to the following key name/value pairs:

- Key=PatchSeverity,Value=Critical
- Key=OS,Value=RHEL
- Key=Environment,Value=Production

13. Choose **Create patch baseline**.

**Create a custom patch baseline (Linux)**

Use the following procedure to create a custom patch baseline for Linux instances. For information about creating a patch baseline for Windows Server instances, see Create a custom patch baseline (Windows) (p. 960).

**To create a custom patch baseline for Linux instances**

2. In the navigation pane, choose **Patch Manager**.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (☰) to open the navigation pane, and then choose **Patch Manager**.
3. Choose **Create patch baseline**.
4. For **Name**, enter a name for your new patch baseline, for example, **MyRHEL PatchBaseline**.
5. (Optional) For **Description**, enter a description for this patch baseline.
6. For **Operating system**, choose an operating system, for example, **Red Hat Enterprise Linux**.
7. If you want to begin using this patch baseline as the default for the selected operating system as soon as you create it, check the box next to **Set this patch baseline as the default patch baseline for operating system name instances**.

   For information about setting an existing patch baseline as the default, see Set an existing patch baseline as the default (p. 965).
8. In the **Approval rules for operating-systems** section, use the fields to create one or more auto-approval rules.

   - **Product**: The version of the operating systems the approval rule applies to, such as **RedhatEnterpriseLinux7.4**. The default selection is **All**.
   - **Classification**: The type of patches the approval rule applies to, such as **Security** or **Enhancement**. The default selection is **All**.
Tip
You can configure a patch baseline to control whether minor version upgrades for Linux are installed, such as RHEL 7.8. Minor version upgrades can be installed automatically by Patch Manager provided that the update is available in the appropriate repository. For Linux operating systems, minor version upgrades are not classified consistently. They can be classified as bug fixes or security updates, or not classified, even within the same kernel version. Here are a few options for controlling whether a patch baseline installs them.

- **Option 1**: The broadest approval rule to ensure minor version upgrades are installed when available is to specify **Classification** as **All** (*) and choose the **Include nonsecurity updates** option.

- **Option 2**: To ensure patches for an operating system version are installed, you can use a wildcard (*) to specify its kernel format in the **Patch exceptions** section of the baseline. For example, the kernel format for RHEL 7.* is *kernel-3.10.0-*.*.el7.*.x86_64*. Enter *kernel-3.10.0-*.*.el7.*.x86_64* in the **Approved patches** list in your patch baseline to ensure all patches, including minor version upgrades, are applied to your RHEL 7.* instances. (If you know the exact package name of a minor version patch, you can enter that instead.)

- **Option 3**: You can have the most control over which patches are applied to your managed instances, including minor version upgrades, by using the **InstallOverrideList (p. 941)** parameter in the **AWS-RunPatchBaseline** document. For more information, see **About the SSM document AWS-RunPatchBaseline (p. 938)**.

- **Severity**: The severity value of patches the rule is to apply to, such as **Critical**. The default selection is **All**.

- **Auto-approval**: The method for selecting patches for automatic approval.

  **Note**
  Because it’s not possible to reliably determine the release dates of update packages for Ubuntu Server, the auto-approval options are not supported for this operating system.

  - **Approve patches after a specified number of days**: The number of days for Patch Manager to wait after a patch is released before a patch is automatically approved. You can enter any integer from zero (0) to 100.

  - **Approve patches released up to a specific date**: The patch release date for which Patch Manager automatically applies all patches released on or before that date. For example, if you specify July 7, 2020, no patches released on or after July 8, 2020, are installed automatically.

  - (Optional) **Compliance reporting**: The severity level you want to assign to patches approved by the baseline, such as **High**.

  **Note**
  If an approved patch is reported as missing, the option you choose in **Compliance reporting**, such as **Critical** or **Medium**, determines the severity of the compliance violation.

  - **Include non-security updates**: Select the check box to install nonsecurity Linux operating system patches made available in the source repository, in addition to the security-related patches.

  **Note**
  For SUSE Linux Enterprise Server, (SLES) it isn’t necessary to select the check box because patches for security and nonsecurity issues are installed by default on SLES instances. For more information, see the content for SLES in **How security patches are selected (p. 913)**.

For more information about working with approval rules in a custom patch baseline, see **About custom baselines (p. 950)**.
9. If you want to explicitly approve any patches in addition to those meeting your approval rules, do the following in the **Patch exceptions** section:

   • For **Approved patches**, enter a comma-separated list of the patches you want to approve.
     
     **Note**
     For information about accepted formats for lists of approved patches and rejected patches, see *About package name formats for approved and rejected patch lists* (p. 952).

   • (Optional) For **Approved patches compliance level**, assign a compliance level to the patches in the list.

   • If any approved patches you specify aren't related to security, select the **Approved patches include non-security updates** box for these patches to be installed on your Linux operating system as well.

10. If you want to explicitly reject any patches that otherwise meet your approval rules, do the following in the **Patch exceptions** section:

   • For **Rejected patches**, enter a comma-separated list of the patches you want to reject.

   **Note**
   For information about accepted formats for lists of approved patches and rejected patches, see *About package name formats for approved and rejected patch lists* (p. 952).

   • For **Rejected patches action**, select the action for Patch Manager to take on patches included in the **Rejected patches** list.

     • **Allow as dependency**: A package in the **Rejected patches** list is installed only if it is a dependency of another package. It is considered compliant with the patch baseline and its status is reported as **InstalledOther**. This is the default action if no option is specified.

     • **Block**: Packages in the **Rejected patches** list, and packages that include them as dependencies, are not installed under any circumstances. If a package was installed before it was added to the **Rejected patches** list, it is considered noncompliant with the patch baseline and its status is reported as **InstalledRejected**.

11. (Optional) If you want to specify alternative patch repositories for different versions of an operating system, such as *AmazonLinux2016.03* and *AmazonLinux2017.09*, do the following for each product in the **Patch sources** section:

   • In **Name**, enter a name to help you identify the source configuration.

   • In **Product**, select the version of the operating systems the patch source repository is for, such as *RedhatEnterpriseLinux7.4*.

   • In **Configuration**, enter the value of the yum repository configuration to use. For example:

     ```
     [main]
     cachedir=/var/cache/yum/$basesearch$releasever
     keepcache=0
     debuglevel=2
     ```

     Choose **Add another source** to specify a source repository for each additional operating system version, up to a maximum of 20.

     For more information about alternative source patch repositories, see *How to specify an alternative patch source repository* (Linux) (p. 917).

12. (Optional) For **Manage tags**, apply one or more tag key name/value pairs to the patch baseline.

   Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a patch baseline to identify the severity level of patches it specifies, the operating system family it
applies to, and the environment type. In this case, you could specify tags similar to the following key name/value pairs:

- Key=PatchSeverity, Value=Critical
- Key=OS, Value=RHEL
- Key=Environment, Value=Production

13. Choose **Create patch baseline**.

### Set an existing patch baseline as the default

When you create a custom patch baseline, you can set the baseline as the default for the associated operating system type as soon as you create it. For information, see [Create a custom patch baseline](p. 959).

You can also set an existing patch baseline as the default for an operating system type.

#### To set a patch baseline as the default

2. In the navigation pane, choose **Patch Manager**.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Patch Manager**.
3. In the patch baselines list, choose the button of a patch baseline that is not currently set as the default for an operating system type.
   **Tip**
   The **Default baseline** column indicates which baselines are currently set as the defaults.
4. In the **Actions** menu, choose **Set default patch baseline**.
5. In the confirmation dialog box, choose **Set default**.

### Create a patch group

To help you organize your patching efforts, we recommend that you add instances to patch groups by using tags. Patch groups require use of the tag key **Patch Group**. You can specify any tag value, but the tag key must be **Patch Group**. For more information about patch groups, see [About patch groups](p. 953).

After you group your instances using tags, you add the patch group value to a patch baseline. By registering the patch group with a patch baseline, you ensure that the correct patches are installed during the patching operation.

#### Topics
- Task 1: Add EC2 instances to a patch group using tags (p. 965)
- Task 2: Add managed instances to a patch group using tags (p. 966)
- Task 3: Add a patch group to a patch baseline (p. 967)

#### Task 1: Add EC2 instances to a patch group using tags

For EC2 instances, you can add tags by using the AWS Systems Manager console, the Amazon EC2 console, the AWS CLI command `create-tags`, or the API action `CreateTags`. 
Note
When using the Amazon EC2 console and AWS CLI, it's possible to apply Key = Patch Group
tags to instances that aren't yet configured for use with Systems Manager. Ensure that SSM
Agent is installed and running on instances that you want to manage using Systems Manager.
For more information, see Working with SSM Agent (p. 63).

To add EC2 instances to a patch group (AWS Systems Manager console)
2. In the navigation pane, choose Managed Instances.
3. In the Managed instances list, choose a the ID of a managed EC2 instance that you want to
   configure for patching.
4. Select the Tags tab, then choose Edit.
5. In the left column, type Patch Group.
6. In the right column, enter a value that helps you understand which instances will be patched.
7. Choose Save.
8. Repeat this procedure to add other managed instances to the same patch group.

To add EC2 instances to a patch group (Amazon EC2 console)
1. Open the Amazon EC2 console, and then choose Instances in the navigation pane.
2. In the list of instances, choose an instance that you want to configure for patching.
3. In the Actions menu, choose Instance Settings, Add/Edit Tags.
4. If the instance already has one or more tags applied, choose Create Tag.
5. For Key, type Patch Group.
6. For Value, enter a value that helps you understand which instances will be patched.
7. Choose Save.
8. Repeat this procedure to add other instances to the same patch group.

To add EC2 instances to a patch group (AWS CLI)
1. Install and configure the AWS CLI, if you have not already.
   For information, see Install or upgrade AWS command line tools (p. 57).
2. Run the following command to add the Patch Group tag to an EC2 instance.

   ```bash
   aws ec2 create-tags --resources "i-1234567890abcdef0" --tags "Key=Patch Group,Value=GroupValue"
   ```

Task 2: Add managed instances to a patch group using tags
For hybrid managed instances (mi-*), you can add tags by using the AWS Systems Manager console, the
AWS CLI command add-tags-to-resource, or the API action AddTagsToResource. You cannot add
tags for hybrid managed instances using the Amazon EC2 console.

To add managed instances to a patch group (AWS Systems Manager console)
2. In the navigation pane, choose Managed Instances.
3. In the Managed instances list, choose a managed instance that you want to configure for patching.
4. Choose **View details**.
5. Select the **Tags** tab, then choose **Edit**.
6. In the left column, type **Patch Group**.
7. In the right column, enter a value that helps you understand which instances will be patched.
8. Choose **Save**.
9. Repeat this procedure to add other managed instances to the same patch group.

**To add managed instances to a patch group (AWS CLI)**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see [Install or upgrade AWS command line tools](p. 57).
2. Run the following command to add the **Patch Group** tag to a managed instance.

   **Linux**
   
   ```bash
   aws ssm add-tags-to-resource \
   --resource-type "ManagedInstance" \
   --resource-id "mi-0123456789abcdefg" \
   --tags "Key=Patch Group,Value=GroupValue"
   ```

   **Windows**
   
   ```bash
   aws ssm add-tags-to-resource ^
   --resource-type "ManagedInstance" ^
   --resource-id "mi-0123456789abcdefg" ^
   --tags "Key=Patch Group,Value=GroupValue"
   ```

**Task 3: Add a patch group to a patch baseline**

To associate a specific patch baseline with your instances, you must add the patch group value to the patch baseline. By registering the patch group with a patch baseline, you can ensure that the correct patches are installed during a patching operation. For more information about patch groups, see [About patch groups](p. 953).

**To add a patch group to a patch baseline (console)**

2. In the navigation pane, choose **Patch Manager**.
3. In the **Patch Baselines** list, choose the patch baseline you want to configure for your patch group.
4. Choose **Actions**, then **Modify patch groups**.
5. Enter the tag value you added to your managed instances in the previous section, then choose **Add**.

**To add a patch group to a patch baseline (AWS CLI)**

1. Install and configure the AWS CLI, if you have not already.
   
   For information, see [Install or upgrade AWS command line tools](p. 57).
2. Run the following command to associate a **Patch Group** tag value to the specified patch baseline.

   **Linux**
   
   ```bash
   aws ssm register-patch-baseline-for-patch-group \
   --patch-group "GroupValue" \
   --patch-baseline "PatchBaselineValue"
   ```
Windows

```bash
aws ssm register-patch-baseline-for-patch-group
   --baseline-id "pb-0123456789abcdef0" ^
   --patch-group "Development"
```

The system returns information like the following:

```json
{
   "PatchGroup": "Development",
   "BaselineId": "pb-0123456789abcdef0"
}
```

### Create a maintenance window for patching

**Important**

You can continue to use this legacy topic to create a maintenance window for patching. However, we recommend using the **Configure patching** page instead. For more information, see Create a patching configuration (console) (p. 971).

To minimize the impact on your server availability, we recommend that you configure a maintenance window to run patching during times that won’t interrupt your business operations. For more information about maintenance windows, see AWS Systems Manager Maintenance Windows (p. 620).

You must configure roles and permissions for Maintenance Windows before beginning this procedure. For more information, see Controlling access to maintenance windows (p. 621).

**To create a maintenance window for patching**

2. In the navigation pane, choose **Maintenance Windows**.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (☰) to open the navigation pane, and then choose **Maintenance Windows**.
3. Choose **Create maintenance window**.
4. For **Name**, enter a name that designates this as a maintenance window for patching critical and important updates.
5. For **Description**, enter a description.
6. Choose **Allow unregistered targets** if you want to allow a maintenance window task to run on managed instances, even if you have not registered those instances as targets. If you choose this option, then you can choose the unregistered instances (by instance ID) when you register a task with the maintenance window. If you don’t choose this option, then you must choose previously-registered targets when you register a task with the maintenance window.
7. In the top of the **Schedule** section, specify a schedule for the maintenance window by using one of the three scheduling options.

   For information about building cron/rate expressions, see Reference: Cron and rate expressions for Systems Manager (p. 1189).
For **Duration**, type the number of hours the maintenance window will run. The value you specify determines the specific end time for the maintenance window based on the time it begins. No maintenance window tasks are permitted to start after the resulting endtime minus the number of hours you specify for **Stop initiating tasks** in the next step.

For example, if the maintenance window starts at 3 PM, the duration is three hours, and the **Stop initiating tasks** value is one hour, no maintenance window tasks can start after 5 PM.

For **Stop initiating tasks**, enter the number of hours before the end of the maintenance window that the system should stop scheduling new tasks to run.

(Optional) For **Start date (optional)**, specify a date and time, in ISO-8601 Extended format, for when you want the maintenance window to become active. This allows you to delay activation of the maintenance window until the specified future date.

(Optional) For **End date (optional)**, specify a date and time, in ISO-8601 Extended format, for when you want the maintenance window to become inactive. This allows you to set a date and time in the future after which the maintenance window no longer runs.

(Optional) For **Time zone (optional)**, specify the time zone to base scheduled maintenance window executions on, in Internet Assigned Numbers Authority (IANA) format. For example: "America/Los_Angeles", "etc/UTC", or "Asia/Seoul".

For more information about valid formats, see the Time Zone Database on the IANA website.

Choose Create maintenance window.

In the maintenance windows list, choose the maintenance window you just created, and then choose Actions, Register targets.

(Optional) In the Maintenance window target details section, provide a name, a description, and owner information (your name or alias) for this target.

For **Targets**, choose Specifying instance tags.

For **Instance tags**, enter a tag key and a tag value to identify the instances to register with the maintenance window, and then choose Add.

Choose Register target. The system creates a maintenance window target.

In the details page of the maintenance window you created, choose Actions, Register Run command task.

(Optional) For Maintenance window task details, provide a name and description for this task.

For **Command document**, choose AWS-RunPatchBaseline.

For **Task priority**, choose a priority. Zero (0) is the highest priority.

For **Targets**, under Target by, choose the maintenance window target you created earlier in this procedure.

(Optional) For **Rate control**:

- For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

  **Note**
  If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

- For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

For **IAM service role**, choose one of the following options to provide permissions for Systems Manager to run tasks on your target instances:

- Create and use a service-linked role for Systems Manager
Service-linked roles provide a secure way to delegate permissions to AWS services because only the linked service can assume a service-linked role. Additionally, AWS automatically defines and sets the permissions of service-linked roles, depending on the actions that the linked service performs on your behalf.

**Note**
If a service-linked role has already been created for your account, choose Use the service-linked role for Systems Manager.

- **Use a custom service role**

You can create a custom service role for maintenance window tasks if you want to use stricter permissions than those provided by the service-linked role.

If you need to create a custom service role, see one of the following topics:
- Control access to maintenance windows (console) (p. 622)
- Control access to maintenance windows (AWS CLI) (p. 626)
- Control access to maintenance windows (Tools for Windows PowerShell) (p. 630)

To help you decide whether to use a custom service role or the Systems Manager service-linked role with a maintenance window task, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).

26. **(Optional) For Output options**, to save the command output to a file, select the Enable writing output to S3 box. Type the bucket and prefix (folder) names in the boxes.

**Note**
The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

To stream the output to a CloudWatch Logs log group, select the CloudWatch output box. Type the log group name in the box.

27. In the SNS notifications section, if you want notifications sent about the status of the command execution, select the Enable SNS notifications check box.

For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

28. **For Parameters:**

- For Operation, choose Scan to scan for missing patches, or choose Install to scan for and install missing patches.

- You don't need to enter anything in the Snapshot Id field. This system automatically generates and provides this parameter.

- You don't need to enter anything in the Install Override List field unless you want Patch Manager to use a different patch set than is specified for the patch baseline. For information, see Parameter name: InstallOverrideList (p. 941).

- For Reboot option, specify whether you want instances to reboot if patches are installed during the Install operation, or if Patch Manager detects other patches that were installed since the last instance reboot. For information, see Parameter name: RebootOption (p. 945).

- (Optional) For Comment, enter a tracking note or reminder about this command.

- For Timeout (seconds), enter the number of seconds the system should wait for the operation to finish before it is considered unsuccessful.

29. Choose Register run command task.
After the maintenance window task completes, you can view patch compliance details in the Systems Manager console on the Managed Instances page. In the filter bar, use the AWS:PatchSummary and AWS:PatchCompliance filters.

**Note**
You can save your query by bookmarking the URL after you specify the filters.

You can also drill down on a specific instance by choosing the instance in the Managed Instances page, and then choosing the Patch tab. You can also use the DescribePatchGroupState and DescribeInstancePatchStatesForPatchGroup APIs to view compliance details. For information about patch compliance data, see About patch compliance (p. 700).

**Create a patching configuration (console)**

A patching configuration defines a unique patching operation. The configuration specifies the instances for patching, which patch baseline is to be applied, the schedule for patching, and the maintenance window that the configuration is to be associated with.

**Note**
Most patching use cases benefit from patching instances on a schedule with a maintenance window, but you can also run a one-time patching operation manually without a maintenance window.

To minimize the impact on your server availability, we recommend that you configure a maintenance window to run patching during times that won't interrupt your business operations. For more information about maintenance windows, see AWS Systems Manager Maintenance Windows (p. 620).

If you plan to add the patching configuration to a maintenance window, you must first configure roles and permissions for Maintenance Windows before beginning this procedure. For more information, see Controlling access to maintenance windows (p. 621).

**To create a patching configuration (console)**

2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Patch Manager.
3. Choose Configure patching.
4. In the **Instances to patch** section, choose one of the following:
   - **Enter instance tags**: Enter a tag key and optional tag value to specify the tagged instance to patch. Click Add to include additional tagged instances.
   - **Select a patch group**: Choose the name of an existing patch group that includes the instances you want to patch.

**Note**
The Select a patch group list displays only those patch groups that are attached to, or registered with, a patch baseline. You can register a patch group with a patch baseline in one of two ways. You can use the register-patch-baseline-for-patch-group CLI command, or you can view a patch baseline in the Systems Manager console and select Modify patch groups from the Actions menu. Alternatively, to specify an existing patch group that is not registered with the patch baseline, choose Enter instance tag, type Patch Group as the tag key and the patch group’s name as the tag value.
AWS Systems Manager User Guide
Working with Patch Manager (console)

- **Select instances manually**: Select the check box next to the name of each instance you want to patch.

5. In the **Patching schedule** section, choose one of the following:

- **Select an existing maintenance window**: From the list, select a maintenance window you have already created, and then continue to step 7.
- **Schedule in a new maintenance window**: Create a new maintenance window to associate with this patching configuration.
- **Skip scheduling and patch now**: Run a one-time manual patching operation without a schedule or maintenance window. Continue to step 7.

6. If you chose **Schedule in a new maintenance window** in step 5, then under **How do you want to specify a patching schedule?**, do the following:

- Under **How do you want to specify a maintenance window schedule?**, choose a schedule builder or expression option.
- Under **maintenance window run frequency**, specify how frequently the maintenance window runs. If you are specifying a CRON/Rate expression, see Reference: Cron and rate expressions for Systems Manager (p. 1189) for more information.
- For **Maintenance window duration**, specify the number of hours the maintenance window is permitted to run before timing out.
- For **Maintenance window name**, enter a name to identify the maintenance window.

7. In the **Patching operation** area, choose whether to scan instances for missing patches and apply them as needed, or to scan only and generate a list of missing patches.

8. (Optional) In the **Additional settings** area, if any target instances you selected belong to a patch group, you can change the patch baseline that is associated with the patch group. To do so, follow these steps:

1. Choose the button next to the name of the associated patch baseline.
2. Choose **Change patch baseline registration**.
3. Choose the patch baselines you want to specify for this configuration by clearing and selecting check boxes next to the patch baseline names.
4. Choose **Close**.

**Note**
For any target instances you selected that are not part of a patch group, Patch Manager instead uses the default patch baseline for the operating system type of the instance.

9. Choose **Configure patching**.

If you created a new maintenance window for this patching configuration, you can add to it or make patching configuration changes in the **Maintenance Windows** area of Systems Manager. For more information, see Update or delete a maintenance window (console) (p. 639).

**Update or delete a patch baseline (console)**

You can update or delete a custom patch baseline that you have created. When you update a patch baseline, you can change its name or description, its approval rules, and its exceptions for approved and rejected patches. You can also update the tags that are applied to the patch baseline. You can't change the operating system type that a patch baseline has been created for, and you can't make changes to a predefined patch baseline provided by AWS.

**Update or delete a patch baseline (console)**

Follow these steps to update or delete a patch baseline.
To update or delete a patch baseline (console)

2. In the navigation pane, choose Patch Manager.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon ( saturn) to open the navigation pane, and then choose Patch Manager.
3. Choose the patch baseline that you want to update or delete, and then do one of the following:
   - To remove the patch baseline from your account, choose Delete. The system prompts you to confirm your actions.
   - To make changes to the patch baseline name or description, approval rules, or patch exceptions, choose Edit. On the Edit patch baseline page, change the values and options that you want, and then choose Save changes.
   - To add, change, or delete tags applied to the patch baseline, choose the Tags tab, and then choose Edit tags. On the Edit patch baseline tags page, make updates to the patch baseline tags, and then choose Save changes.

For information about the configuration choices you can make, see Create a custom patch baseline (p. 959).

Working with Patch Manager (AWS CLI)

The section includes examples of CLI commands that you can use to perform Patch Manager configuration tasks.

For an illustration of using the AWS CLI to patch a server environment by using a custom patch baseline, see Walkthrough: Patch a server environment (command line) (p. 991).

For more information about using the CLI for AWS Systems Manager tasks, see the AWS Systems Manager section of the AWS CLI Command Reference.

Sample commands
- Create a patch baseline (p. 974)
- Create a patch baseline with custom repositories for different OS versions (p. 975)
- Update a patch baseline (p. 976)
- Rename a patch baseline (p. 977)
- Delete a patch baseline (p. 978)
- List all patch baselines (p. 978)
- List all AWS-provided patch baselines (p. 979)
- List my patch baselines (p. 980)
- Display a patch baseline (p. 980)
- Get the default patch baseline (p. 981)
- Set a custom patch baseline as the default (p. 982)
- Reset an AWS patch baseline as the default (p. 982)
- Register a patch group "web servers" with a patch baseline (p. 982)
- Register a patch group "Backend" with the AWS-provided patch baseline (p. 983)
- Display patch group registrations (p. 983)
- Deregister a patch group from a patch baseline (p. 984)
Create a patch baseline

The following command creates a patch baseline that approves all critical and important security updates for Windows Server 2012 R2 five days after they are released. Patches have also been specified for the Approved and Rejected patch lists. In addition, the patch baseline has been tagged to indicate that it is for a production environment.

Linux

```bash
aws ssm create-patch-baseline
   --name "Windows-Server-2012R2"
   --tags "Key=Environment,Value=Production"
   --description "Windows Server 2012 R2, Important and Critical security updates"
   --approved-patches "KB2032276,MS10-048"
   --rejected-patches "KB2124261"
   --rejected-patches-action "ALLOW_AS_DEPENDENCY"
   --approval-rules
   "PatchRules=([{PatchFilterGroup={PatchFilters=[{"Key=MSRC_SEVERITY,Values=[Important,Critical]},
   {Key=CLASSIFICATION,Values=SecurityUpdates},
   {Key=PRODUCT,Values=WindowsServer2012R2}]}],ApproveAfterDays=5}]
```

Windows

```bash
aws ssm create-patch-baseline
   --name "Windows-Server-2012R2"
   --tags "Key=Environment,Value=Production"
   --description "Windows Server 2012 R2, Important and Critical security updates"
   --approved-patches "KB2032276,MS10-048"
   --rejected-patches "KB2124261"
   --rejected-patches-action "ALLOW_AS_DEPENDENCY"
   --approval-rules
   "PatchRules=([{PatchFilterGroup={PatchFilters=[{"Key=MSRC_SEVERITY,Values=[Important,Critical]},
   {Key=CLASSIFICATION,Values=SecurityUpdates},
   {Key=PRODUCT,Values=WindowsServer2012R2}]}],ApproveAfterDays=5}]
```

The system returns information like the following.

```json
{
   "BaselineId": "pb-0c10e65780EXAMPLE"
}
```
Create a patch baseline with custom repositories for different OS versions

Applies to Linux instances only. The following command shows how to specify the patch repository to use for a particular version of the Amazon Linux operating system. This sample uses a source repository enabled by default on Amazon Linux 2017.09, but could be adapted to a different source repository that you have configured for an instance.

**Note**
To better demonstrate this more complex command, we are using the `--cli-input-json` option with additional options stored in an external JSON file.

1. Create a JSON file with a name like `my-patch-repository.json` and add the following content to it:

```json
{
  "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\nmirrorlist=http://repo./$awsregion./$awsdomain./$releasever/main/mirror.list //nmirrorlist_expire=300//nmetadata_expire=300
npriority=10 \nfailovermethod=priority \nfastestmirror_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. In the directory where you saved the file, run the following command:

```
aws ssm create-patch-baseline --cli-input-json file://my-patch-repository.json
```

The system returns information like the following:

```
{
  "BaselineId": "pb-12343b962ba63wxya"
}
```

## Update a patch baseline

The following command adds two patches as rejected and one patch as approved to an existing patch baseline.

### Note

For information about accepted formats for lists of approved patches and rejected patches, see
About package name formats for approved and rejected patch lists (p. 952).

**Linux**

```
aws ssm update-patch-baseline \
  --baseline-id pb-0c10e65780EXAMPLE \
  --rejected-patches "KB2032276" "MS10-048" \
  --approved-patches "KB2124261"
```

**Windows**

```
aws ssm update-patch-baseline ^
  --baseline-id pb-0c10e65780EXAMPLE ^
  --rejected-patches "KB2032276" "MS10-048" ^
  --approved-patches "KB2124261"
```

The system returns information like the following.

```
{
  "BaselineId":"pb-0c10e65780EXAMPLE",
  "Name":"Windows-Server-2012R2",
  "RejectedPatches":[
    "KB2032276",
    "MS10-048"
  ],
  "GlobalFilters":{
    "PatchFilters":[
    
  ]
  },
  "ApprovalRules":{
    "PatchRules":[
    
  ]
  }
}
```
"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"
}

### Rename a patch baseline

**Linux**

```bash
aws ssm update-patch-baseline \
  --baseline-id pb-0c10e65780EXAMPLE \
  --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"
```

**Windows**

```bash
aws ssm update-patch-baseline ^
  --baseline-id pb-0c10e65780EXAMPLE ^
  --name "Windows-Server-2012-R2-Important-and-Critical-Security-Updates"
```

The system returns information like the following.

```json
{
  "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,  "CreatedDate": 1480997823.81,  "ApprovedPatches": [  "KB2124261"  ],  "Description": "Windows Server 2012 R2, Important and Critical security updates"  }

Delete a patch baseline

aws ssm delete-patch-baseline --baseline-id "pb-0c10e65780EXAMPLE"

The system returns information like the following.

{
  "BaselineId": "pb-0c10e65780EXAMPLE"
}

List all patch baselines

aws ssm describe-patch-baselines

The system returns information like the following.

{
  "BaselineIdentities": [  {  "BaselineName": "AWS-DefaultPatchBaseline",  "DefaultBaseline": true,  "BaselineDescription": "Default Patch Baseline Provided by AWS.",  "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
  }  ]
}
Here is another command that lists all patch baselines in a Region.

Linux

```bash
aws ssm describe-patch-baselines \
  --region us-east-2 \
  --filters "Key=OWNER,Values=[All]"
```

Windows

```bash
aws ssm describe-patch-baselines ^ \
  --region us-east-2 ^ \
  --filters "Key=OWNER,Values=[All]"
```

The system returns information like the following.

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

**List all AWS-provided patch baselines**

Linux

```bash
aws ssm describe-patch-baselines \
  --region us-east-2 \
  --filters "Key=OWNER,Values=[AWS]"
```

Windows

```bash
aws ssm describe-patch-baselines ^ \
  --region us-east-2 ^
```
--filters "Key=OWNER,Values=[AWS]"

The system returns information like the following.

```
{
  "BaselineIdentities": [
    {
      "BaselineName": "AWS-DefaultPatchBaseline",
      "DefaultBaseline": true,
      "BaselineDescription": "Default Patch Baseline Provided by AWS.",
      "BaselineId": "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
    }
  ]
}
```

**List my patch baselines**

**Linux**

```
aws ssm describe-patch-baselines \
  --region us-east-2 \
  --filters "Key=OWNER,Values=[Self]"
```

**Windows**

```
aws ssm describe-patch-baselines ^
  --region us-east-2 ^
  --filters "Key=OWNER,Values=[Self]"
```

The system returns information like the following.

```
{
  "BaselineIdentities": [
    {
      "BaselineName": "Windows-Server-2012R2",
      "DefaultBaseline": false,
      "BaselineDescription": "Windows Server 2012 R2, Important and Critical security updates",
      "BaselineId": "pb-0c10e65780EXAMPLE"
    }
  ]
}
```

**Display a patch baseline**

```
aws ssm get-patch-baseline --baseline-id pb-0c10e65780EXAMPLE
```

**Note**

For custom patch baselines, you can specify either the patch baseline ID or the full ARN. For AWS-provided patch baseline, you must specify the full ARN. For example, `arn:aws:ssm:us-east-1:075727635805:patchbaseline/pb-03e3f588eec25344c`.

The system returns information like the following.
Get the default patch baseline

```
aws ssm get-default-patch-baseline --region us-east-2
```

The system returns information like the following.

```
{
  "BaselineId":"arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
}
```
Set a custom patch baseline as the default

**Linux**

```
aws ssm register-default-patch-baseline \
  --region us-east-2 \
  --baseline-id "pb-0c10e65780EXAMPLE"
```

**Windows**

```
aws ssm register-default-patch-baseline ^
  --region us-east-2 ^
  --baseline-id "pb-0c10e65780EXAMPLE"
```

The system returns information like the following:

```
{
  "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

Reset an AWS patch baseline as the default

**Linux**

```
aws ssm register-default-patch-baseline \
  --region us-east-2 \
  --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/\n  pb-0574b43a65ea646ed"
```

**Windows**

```
aws ssm register-default-patch-baseline ^
  --region us-east-2 ^
  --baseline-id "arn:aws:ssm:us-east-2:733109147000:patchbaseline/\n  pb-0574b43a65ea646ed"
```

The system returns information like the following:

```
{
  "BaselineId": "pb-0c10e65780EXAMPLE"
}
```

Register a patch group "web servers" with a patch baseline

**Linux**

```
aws ssm register-patch-baseline-for-patch-group \
  --baseline-id "pb-0c10e65780EXAMPLE" \
  --patch-group "Web Servers"
```

**Windows**

```
aws ssm register-patch-baseline-for-patch-group ^
  --baseline-id "pb-0c10e65780EXAMPLE" ^
  --patch-group "Web Servers"
```
Register a patch group "Backend" with the AWS-provided patch baseline

Linux

```bash
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"
```

Windows

```bash
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"
```

Display patch group registrations

```bash
aws ssm describe-patch-groups --region us-east-2
```

The system returns information like the following.

```
{
    "PatchGroup":"Backend",
    "BaselineId":"arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
}
```
"BaselineIdentity":{
  "BaselineName":"Windows-Server-2012R2",
  "DefaultBaseline":true,
  "BaselineDescription":"Windows Server 2012 R2, Important and Critical updates",
  "BaselineId":"pb-0c10e65780EXAMPLE"
}

Deregister a patch group from a patch baseline

Linux

```bash
aws ssm deregister-patch-baseline-for-patch-group \
  --region us-east-2 \n  --patch-group "Production" \n  --baseline-id "arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
```

Windows

```bash
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"
```

The system returns information like the following.

```
{
  "PatchGroup":"Production",
  "BaselineId":"arn:aws:ssm:us-east-2:111122223333:patchbaseline/pb-0c10e65780EXAMPLE"
}
```

Get all patches defined by a patch baseline

Linux

*This command is supported for Windows Server patch baselines only.*

Windows

```bash
aws ssm describe-effective-patches-for-patch-baseline ^ 
  --region us-east-2 ^ 
  --baseline-id "pb-0c10e65780EXAMPLE"
```

The system returns information like the following.

```
{
  "NextToken":"--token string truncated--",
  "EffectivePatches":[
    {
      "PatchStatus":{
        "ApprovalDate":1384711200.0,
        "DeploymentStatus":"APPROVED"
      },
      "Patch":{
```
Get all patches for Windows Server 2012 that have a MSRC severity of Critical

Linux

```bash
aws ssm describe-available-patches \
  --region us-east-2 \
  --filters Key=PRODUCT,Values=WindowsServer2012 Key=MSRC_SEVERITY,Values=Critical
```

Windows

```bash
aws ssm describe-available-patches ^
```
AWS Systems Manager User Guide
Working with Patch Manager (AWS CLI)

--region us-east-2
--filters Key=PRODUCT,Values=WindowsServer2012 Key=MSRC_SEVERITY,Values=Critical

The system returns information like the following.

```
{"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---
```

Get all available patches

```
aws ssm describe-available-patches --region us-east-2
```

The system returns information like the following.

```
{
  "NextToken": "--token string truncated--",
  "Patches":[
    {
      "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 (KB2032276)",
      "ReleaseDate":1352829600.0,
      "MsrcClassification":"Critical",
      "Language":"All",
      "KbNumber":null,
      "MsrcNumber":null,
      "Id":null
    }
  ]
```

986
Tag a patch baseline

**Linux**

```
aws ssm add-tags-to-resource 
  --resource-type "PatchBaseline" 
  --resource-id "pb-0c10e65780EXAMPLE" 
  --tags "Key=Project,Value=Testing"
```

**Windows**

```
aws ssm add-tags-to-resource ^
  --resource-type "PatchBaseline" ^
  --resource-id "pb-0c10e65780EXAMPLE" ^
  --tags "Key=Project,Value=Testing"
```

List the tags for a patch baseline

**Linux**

```
aws ssm list-tags-for-resource 
  --resource-type "PatchBaseline" 
```
Windows

```
aws ssm list-tags-for-resource
  --resource-id "pb-0c10e65780EXAMPLE"
```

Remove a tag from a patch baseline

Linux

```
aws ssm remove-tags-from-resource
  --resource-type "PatchBaseline"
  --resource-id "pb-0c10e65780EXAMPLE"
  --tag-keys "Project"
```

Windows

```
aws ssm remove-tags-from-resource
  --resource-type "PatchBaseline"
  --resource-id "pb-0c10e65780EXAMPLE"
  --tag-keys "Project"
```

Get patch summary states per-instance

The per-instance summary gives you a number of patches in the following states per instance: "NotApplicable", "Missing", "Failed", "InstalledOther" and "Installed".

Linux

```
aws ssm describe-instance-patch-states
  --instance-ids i-08ee91c0b17045407 i-09a618aec652973a9
```

Windows

```
aws ssm describe-instance-patch-states
  --instance-ids i-08ee91c0b17045407 i-09a618aec652973a9
```

The system returns information like the following.

```
{
  "InstancePatchStates": {
  {
    "InstanceId": "i-08ee91c0b17045407",
    "PatchGroup": "",
    "BaselineId": "pb-0e392de35e7c563b7",
    "SnapshotId": "6d03d6c5-f79d-41d0-8d0e-00a9aEXAMPLE",
    "InstalledCount": 50,
    "InstalledOtherCount": 353,
    "InstalledPendingRebootCount": 0,
    "InstalledRejectedCount": 0,
    "MissingCount": 0,
  }
}
```

988
"FailedCount": 0,
"UnreportedNotApplicableCount": -1,
"NotApplicableCount": 671,
"OperationStartTime": "2020-01-24T12:37:56-08:00",
"OperationEndTime": "2020-01-24T12:37:59-08:00",
"Operation": "Scan",
"RebootOption": "NoReboot"
},
{
"InstanceId": "i-09a618aec652973a9",
"PatchGroup": "",
"BaselineId": "pb-07e64d4e9bc703f2e3",
"SnapshotId": "c7e0441b-1eae-411b-8aa7-973e6EXAMPLE",
"InstalledCount": 36,
"InstalledOtherCount": 396,
"InstalledPendingRebootCount": 0,
"InstalledRejectedCount": 0,
"MissingCount": 3,
"FailedCount": 0,
"UnreportedNotApplicableCount": -1,
"NotApplicableCount": 420,
"OperationStartTime": "2020-01-24T12:37:34-08:00",
"OperationEndTime": "2020-01-24T12:37:37-08:00",
"Operation": "Scan",
"RebootOption": "NoReboot"
}

---output truncated---

Get patch compliance details for an instance

aws ssm describe-instance-patches --instance-id i-08ee91c0b17045407

The system returns information like the following.

```json
{
    "NextToken":"--token string truncated--",
    "Patches": [
        {
            "Title": "bind-libs.x86_64:32:9.8.2-0.68.rc1.60.amzn1",
            "KBId": "bind-libs.x86_64",
            "Classification": "Security",
            "Severity": "Important",
            "State": "Installed",
            "InstalledTime": "2019-08-26T11:05:24-07:00"
        },
        {
            "Title": "bind-utils.x86_64:32:9.8.2-0.68.rc1.60.amzn1",
            "KBId": "bind-utils.x86_64",
            "Classification": "Security",
            "Severity": "Important",
            "State": "Installed",
            "InstalledTime": "2019-08-26T11:05:32-07:00"
        },
        {
            "Title": "dhclient.x86_64:12:4.1.1-53.F1.28.amzn1",
            "KBId": "dhclient.x86_64",
            "Classification": "Security",
            "Severity": "Important",
            "State": "Installed",
            "InstalledTime": "2019-08-26T11:05:31-07:00"
        }
    ]
}
---output truncated---

989
AWS Systems Manager Patch Manager walkthroughs

The walkthroughs in this section demonstrate how to use Patch Manager for a selection of patching scenarios.

Topics
- Walkthrough: Create a patch baseline for installing Windows Service Packs (console) (p. 990)
- Walkthrough: Patch a server environment (command line) (p. 991)

Walkthrough: Create a patch baseline for installing Windows Service Packs (console)

When you create a custom patch baseline, you can specify that all, some, or only one type of supported patch is installed.

In patch baselines for Windows, you can select ServicePacks as the only Classification option in order to limit patching updates to Service Packs only. Service Packs can be installed automatically by Patch Manager provided that the update is available in Windows Update or Windows Server Update Services (WSUS).

You can configure a patch baseline to control whether Service Packs for all Windows versions are installed, or just those for specific versions, such as Windows 7 or Windows Server 2016.

Use the following procedure to create a custom patch baseline to be used exclusively for installing all Service Packs on your Windows instances.

To use Patch Manager to install Windows Service Packs (console) (Windows)

2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Patch Manager.
3. Choose Create patch baseline.
4. For Name, enter a name for your new patch baseline, for example, MyWindowsServicePackPatchBaseline.
5. (Optional) For Description, enter a description for this patch baseline.
6. For Operating system, choose Windows.
7. If you want to begin using this patch baseline as the default for Windows as soon as you create it, select Set this patch baseline as the default patch baseline for Windows Server instances.
   If you choose not to set this patch baseline for use now, you can do so later. For information, see Set an existing patch baseline as the default (p. 965).
8. In the Approval rules for operating systems section, use the fields to create one or more auto-approval rules.
   - Product: The operating system versions that the approval rule applies to, such as WindowsServer2008. You can choose one, more than one, or all supported versions of Windows. The default selection is All.
   - Classification: Choose ServicePacks.
   - Severity: The severity value of patches the rule is to apply to. To ensure that all Service Packs are included by the rule, choose All.
• **Auto-approval**: The method for selecting patches for automatic approval.
• **Approve patches after a specified number of days**: The number of days for Patch Manager to wait after a patch is released before a patch is automatically approved. You can enter any integer from zero (0) to 100.
• **Approve patches released up to a specific date**: The patch release date for which Patch Manager automatically applies all patches released on or before that date. For example, if you specify July 7, 2020, no patches released on or after July 8, 2020, are installed automatically.
• (Optional) **Compliance reporting**: The severity level you want to assign to Service Packs approved by the baseline, such as High.

**Note**
If an approved Service Pack is reported as missing, the option you choose in Compliance reporting, such as Critical or Medium, determines the severity of the compliance violation.

9. (Optional) For **Manage tags**, apply one or more tag key name/value pairs to the patch baseline.

   Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For this patch baseline dedicated to updating Service Packs, you could specify key-value pairs such as the following:

   • Key=OS,Value=Windows
   • Key=Classification,Value=ServicePacks

10. Choose **Create patch baseline**.

**Walkthrough: Patch a server environment (command line)**

The following procedure describes how to patch a server environment by using a custom patch baseline, patch groups, and a maintenance window.

**Before you begin**

• Install or update the SSM Agent on your instances. To patch Linux instances, your instances must be running SSM Agent version 2.0.834.0 or later. For more information, see Update SSM Agent by using Run Command (p. 835).
• Configure roles and permissions for the Maintenance Windows capability. For more information, see Controlling access to maintenance windows (p. 621).
• Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already.

   For information, see Install or upgrade AWS command line tools (p. 57).

**To configure Patch Manager and patch instances (command line)**

1. Run the following command to create a patch baseline for Windows named Production-Baseline. This patch baseline approves patches for a production environment seven days after they are released. That is, we have tagged the patch baseline is tagged to indicate that it is for a production environment.

   **Note**
   The OperatingSystem parameter and PatchFilters vary depending on the operating system of the instances the patch baseline applies to. For more information, see OperatingSystem and PatchFilter.

   **Linux**

   ```bash
   aws ssm create-patch-baseline \
   ```
--name "Production-Baseline" \
--operating-system "WINDOWS" \
--tags "Key=Environment,Value=Production" \
--approval-rules
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important]},
  {Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,ServicePacks,UpdateRollups,CriticalUpdates]}
  ],ApproveAfterDays=7}]

--description "Baseline containing all updates approved for production systems"

Windows

aws ssm create-patch-baseline ^
  --name "Production-Baseline" ^
  --operating-system "WINDOWS" ^
  --tags "Key=Environment,Value=Production" ^
  --approval-rules
  "PatchRules=[{PatchFilterGroup={PatchFilters=[{Key=MSRC_SEVERITY,Values=[Critical,Important]},
  {Key=CLASSIFICATION,Values=[SecurityUpdates,Updates,ServicePacks,UpdateRollups,CriticalUpdates]}
  ],ApproveAfterDays=7}]

  --description "Baseline containing all updates approved for production systems"

PowerShell

$patchApprovalRulesFilterGroup = New-Object
Amazon.SimpleSystemsManagement.Model.PatchFilterGroup
$severityFilter.Key = "MSRC_SEVERITY"
$severityFilter.Values.Add("Critical")
$severityFilter.Values.Add("Important")
$classificationFilter.Key = "CLASSIFICATION"
$classificationFilter.Values.Add("SecurityUpdates")
$classificationFilter.Values.Add("Updates")
$classificationFilter.Values.Add("ServicePacks")
$classificationFilter.Values.Add("UpdateRollups")
$classificationFilter.Values.Add("CriticalUpdates")

$patchApprovalRulesFilterGroup.PatchFilters.Add($severityFilter)
$patchApprovalRulesFilterGroup.PatchFilters.Add($classificationFilter)

$patchApprovalRules.PatchFilterGroup = $patchApprovalRulesFilterGroup
$patchApprovalRules.ApproveAfterDays = 7

New-SSMPatchBaseline `  
  -Name "Production-Baseline"  
  -OperatingSystem "WINDOWS"  
  -Tags @{Key="Environment";Value="Production"}
  -ApprovalRules_PatchRule $patchApprovalRules
  -Description "Baseline containing all updates approved for production systems"

The system returns information like the following.

{
  "BaselineId": "pb-0c10e65780EXAMPLE"
2. Run the following commands to register the "Production-Baseline" patch baseline for two patch
groups. The groups are named "Database Servers" and "Front-End Servers".

Linux

```
aws ssm register-patch-baseline-for-patch-group `  
  --baseline-id pb-0c10e65780EXAMPLE `  
  --patch-group "Database Servers"
```

Windows

```
aws ssm register-patch-baseline-for-patch-group `  
  --baseline-id pb-0c10e65780EXAMPLE `  
  --patch-group "Database Servers"
```

PowerShell

```
Register-SSMPatchBaselineForPatchGroup `  
  -BaselineId pb-0c10e65780EXAMPLE `  
  -PatchGroup "Database Servers"
```

The system returns information like the following.

```json
{
  "PatchGroup":"Database Servers",
  "BaselineId":"pb-0c10e65780EXAMPLE"
}
```

Linux

```
aws ssm register-patch-baseline-for-patch-group `  
  --baseline-id pb-0c10e65780EXAMPLE `  
  --patch-group "Front-End Servers"
```

Windows

```
aws ssm register-patch-baseline-for-patch-group `  
  --baseline-id pb-0c10e65780EXAMPLE `  
  --patch-group "Front-End Servers"
```

PowerShell

```
Register-SSMPatchBaselineForPatchGroup `  
  -BaselineId pb-0c10e65780EXAMPLE `  
  -PatchGroup "Front-End Servers"
```

The system returns information like the following.

```json
{
  "PatchGroup":"Front-End Servers",
  "BaselineId":"pb-0c10e65780EXAMPLE"
}```
3. Run the following commands to create two maintenance windows for the production servers. The first window runs every Tuesday at 10 PM. The second window runs every Saturday at 10 PM. In addition, the maintenance window is tagged to indicate that it is for a production environment.

**Linux**

```
aws ssm create-maintenance-window \
  --name "Production-Tuesdays" \
  --tags "Key=Environment,Value=Production" \
  --schedule "cron(0 0 22 ? * TUE *)" \
  --duration 1 \
  --cutoff 0 \
  --no-allow-unassociated-targets
```

**Windows**

```
aws ssm create-maintenance-window ^
  --name "Production-Tuesdays" ^
  --tags "Key=Environment,Value=Production" ^
  --schedule "cron(0 0 22 ? * TUE *)" ^
  --duration 1 ^
  --cutoff 0 ^
  --no-allow-unassociated-targets
```

**PowerShell**

```
New-SSMMaintenanceWindow `\n  -Name "Production-Tuesdays" `\n  -Tags @"{Key="Environment";Value="Production"}" `\n  -Schedule "cron(0 0 22 ? * TUE *)" `\n  -Duration 1 `\n  -Cutoff 0 `\n  -AllowUnassociatedTarget $false
```

The system returns information like the following.

```
{
  "WindowId": "mw-0c66948c711a3b5bd"
}
```

**Linux**

```
aws ssm create-maintenance-window \
  --name "Production-Saturdays" \
  --tags "Key=Environment,Value=Production" \
  --schedule "cron(0 0 22 ? * SAT *)" \
  --duration 2 \
  --cutoff 0 \
  --no-allow-unassociated-targets
```

**Windows**

```
aws ssm create-maintenance-window ^
  --name "Production-Saturdays" ^
  --tags "Key=Environment,Value=Production" ^
```

---

994
4. Run the following commands to register the Database and Front-End servers patch groups with their respective maintenance windows.

**Linux**

```bash
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"
```

**Windows**

```bash
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"
```

**PowerShell**

```powershell
Register-SSMTargetWithMaintenanceWindow
-WindowId "mw-09e2a75baadd84e85"
-Targets @(Key="tag:Patch Group",Values="Database Servers")
-OwnerInformation "Database Servers"
-ResourceType "INSTANCE"
```

The system returns information like the following.

```
{
  "WindowTargetId": "557e7b3a-bc2f-48dd-ae05-e282b5b20760"
}
```
Linux

aws ssm register-target-with-maintenance-window \
--window-id mw-0c66948c711a3b5bd \
--targets "Key=tag:Patch Group,Values=Front-End Servers" \
--owner-information "Front-End Servers" \
--resource-type "INSTANCE"

Windows

aws ssm register-target-with-maintenance-window ^
--window-id mw-0c66948c711a3b5bd ^
--targets "Key=tag:Patch Group,Values=Front-End Servers" ^
--owner-information "Front-End Servers" ^
--resource-type "INSTANCE"

PowerShell

Register-SSMTargetWithMaintenanceWindow ` 
-WindowId "mw-0c66948c711a3b5bd" ` 
-Targets (@{Key="tag:Patch Group";Values="Front-End Servers"}) ` 
-OwnerInformation "Front-End Servers" ` 
-ResourceType "INSTANCE"

The system returns information like the following.

```
{
  "WindowTargetId":"faa01c41-1d57-496c-ba77-ff9cadba4b7d"
}
```

5. Run the following commands to register a patch task that installs missing updates on the Database and Front-End servers during their respective maintenance windows.

Linux

aws ssm register-task-with-maintenance-window \
--window-id mw-09e2a75baadd84e85 \
--targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-e282b5b20760" \
--task-arn "AWS-RunPatchBaseline" \
--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"]}}'

Windows

aws ssm register-task-with-maintenance-window ^
--window-id mw-09e2a75baadd84e85 ^
--targets "Key=WindowTargetIds,Values=557e7b3a-bc2f-48dd-e282b5b20760" ^
--task-arn "AWS-RunPatchBaseline" ^
--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"]}}'

PowerShell

```powershell
$taskParameters = @{} $taskParameterValues = New-Object Amazon.SimpleSystemsManagement.Model.MaintenanceWindowTaskParameterValueExpression $taskParameterValues.Values = @("Install") $taskParameters.Add("Operation", $taskParameterValues) Register-SSMTaskWithMaintenanceWindow `-WindowId "mw-09e2a75baadd84e85" `-Targets @{Key="WindowTargetIds";Values="557e7b3a-bc2f-48dd-ae05-e282b5b20760"} `-TaskArn "AWS-RunPatchBaseline" `-TaskType "RUN_COMMAND" `-MaxConcurrency 2 `-MaxErrors 1 `-Priority 1 `-TaskParameters $taskParameters

The system returns information like the following.

```json
{
  "WindowTaskId": "968e3b17-8591-4fb2-932a-b62389d6f635"
}
```

Linux

```bash
aws ssm register-task-with-maintenance-window 
--window-id mw-09e2a75baadd84e85 
--targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" 
--task-arn "AWS-RunPatchBaseline" 
--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"]}}'
```

Windows

```bash
aws ssm register-task-with-maintenance-window ^ 
--window-id mw-09e2a75baadd84e85 ^ 
--targets "Key=WindowTargetIds,Values=767b6508-f4ac-445e-b6fe-758cc912e55c" ^ 
--task-arn "AWS-RunPatchBaseline" ^ 
--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"]}}'
```

PowerShell

```powershell
$taskParameters = @()
```
$taskParameterValues.Values = @("Install")
$taskParameters.Add("Operation", $taskParameterValues)

Register-SSMTaskWithMaintenanceWindow `-WindowId "mw-0c66946c711a3b5bd" `-Targets @{Key="WindowTargetIds";Values="faa01c41-1d57-496c-ba77-ff9cadba4b7d"}

-TaskArn "AWS-RunPatchBaseline" `-TaskType "RUN_COMMAND" `-MaxConcurrency 2 `-MaxErrors 1 `-Priority 1 `-TaskParameters $taskParameters

The system returns information like the following.

```json
{
  "WindowTaskId": "09f2e873-a3a7-443f-ba0a-05cf4de5a1c7"
}
```

6. Run the following command to get the high-level patch compliance summary for a patch group. The high-level patch compliance summary includes the number of instances with patches in the respective patch states.

**Note**

It is expected to see zeroes for the number of instances in the summary until the patch task runs during the first maintenance window.

**Linux**

```bash
aws ssm describe-patch-group-state --patch-group "Database Servers"
```

**Windows**

```bash
aws ssm describe-patch-group-state --patch-group "Database Servers"
```

**PowerShell**

```powershell
Get-SSMPatchGroupState -PatchGroup "Database Servers"
```

The system returns information like the following.

```json
{
  "Instances": number,
  "InstancesWithFailedPatches": number,
  "InstancesWithInstalledOtherPatches": number,
  "InstancesWithInstalledPatches": number,
  "InstancesWithInstalledPendingRebootPatches": number,
  "InstancesWithInstalledRejectedPatches": number,
  "InstancesWithMissingPatches": number,
  "InstancesWithNotApplicablePatches": number,
  "InstancesWithUnreportedNotApplicablePatches": number
}
```
7. Run the following command to get patch summary states per-instance for a patch group. The per-instance summary includes a number of patches in the respective patch states per instance for a patch group.

Linux

```
aws ssm describe-instance-patch-states-for-patch-group --patch-group "Database Servers"
```

Windows

```
aws ssm describe-instance-patch-states-for-patch-group --patch-group "Database Servers"
```

PowerShell

```
Get-SSMInstancePatchStatesForPatchGroup -PatchGroup "Database Servers"
```

The system returns information like the following.

```
{
    "InstancePatchStates": [
        {
            "BaselineId": "string",
            "FailedCount": number,
            "InstalledCount": number,
            "InstalledOtherCount": number,
            "InstalledPendingRebootCount": number,
            "InstalledRejectedCount": number,
            "InstallOverrideList": "string",
            "InstanceId": "string",
            "LastNoRebootInstallOperationTime": number,
            "MissingCount": number,
            "NotApplicableCount": number,
            "Operation": "string",
            "OperationEndTime": number,
            "OperationStartTime": number,
            "OwnerInformation": "string",
            "PatchGroup": "string",
            "RebootOption": "string",
            "SnapshotId": "string",
            "UnreportedNotApplicableCount": number
        }
    ]
}
```

For examples of other AWS CLI commands you can use for your Patch Manager configuration tasks, see Working with Patch Manager (AWS CLI) (p. 973).

AWS Systems Manager Distributor

AWS Systems Manager Distributor lets you package your own software—or find AWS-provided agent software packages, such as AmazonCloudWatchAgent—to install on AWS Systems Manager managed instances. Distributor publishes resources, such as software packages, to AWS Systems Manager managed instances. Publishing a package advertises specific versions of the package's document—a
How can Distributor benefit my organization?

Distributor offers these benefits:

• **One package, many platforms**
  One document can have attached .zip files that are installed on different operating systems (such as Windows, Ubuntu Server, Debian, or Red Hat Enterprise Linux). For more information about supported platforms, see Supported package platforms and architectures (p. 1001).

• **Control package access across groups of managed instances**
  You can use Run Command or State Manager to control which of your managed instances get a package and which version of that package. Managed instances can be grouped by instance IDs, AWS account numbers, tags, or AWS Regions. You can use State Manager associations to deliver different versions of a package to different groups of instances.

• **Many AWS agent packages included and ready to use**
  Distributor includes many AWS agent packages that are ready for you to deploy to managed instances. Look for packages in the Distributor Packages list page that are published by Amazon. Examples include AmazonCloudWatchAgent and AWSPVDriver.

• **Automate deployment**
  To keep your environment current, use State Manager to schedule packages for automatic deployment on target instances when those instances are first launched.

Who should use Distributor?

• Any AWS customer who wants to create new or deploy existing software packages, including AWS-published packages, to multiple AWS Systems Manager managed instances at one time.

• Software developers who create software packages.

• Administrators who are responsible for keeping AWS Systems Manager managed instances current with the most up-to-date software packages.

What are the features of Distributor?

• **Deployment of packages to both Windows and Linux instances**
  Distributor lets you deploy software packages to EC2 instances for Linux and Windows Server. For a list of supported instance operating system types, see the section called “Supported package platforms and architectures” (p. 1001).

• **Deploy packages one time, or on an automated schedule**
  You can choose to deploy packages one time, on a regular schedule, or whenever the default package version is changed to a different version.
• **Completely reinstall packages, or perform in-place updates**

To install a new package version, you can completely uninstall the current version and install a new one in its place, or only update the current version with new and updated components, according to an *update script* that you provide. Your package application is unavailable during a reinstallation, but can remain available during an in-place update. In-place updates are especially useful for security monitoring applications or other scenarios where you need to avoid application downtime.

• **Console, CLI, PowerShell, and SDK access to Distributor capabilities**

You can work with Distributor by using the AWS Systems Manager console, AWS CLI, AWS Tools for PowerShell, or the AWS SDK of your choice.

• **IAM access control**

By using IAM policies, you can control which members of your organization can create, update, deploy, or delete packages or package versions. For example, you might want to give an administrator permissions to deploy packages, but not to change packages or create new package versions.

• **Logging and auditing capability support**

You can audit and log Distributor user actions in your AWS account through integration with other AWS services. For more information, see Auditing and logging Distributor activity (p. 1029).

**What is a package?**

A *package* is a collection of installable software or assets that includes the following.

• A .zip file of software per target operating system platform. Each .zip file must include the following.  
  • An *install* and an *uninstall* script. Windows Server-based instances require PowerShell scripts (scripts named *install.ps1* and *uninstall.ps1*). Linux-based instances require shell scripts (scripts named *install.sh* and *uninstall.sh*). SSM Agent reads and carries out the instructions in the *install* and *uninstall* scripts.  
  • An executable file. SSM Agent must find this executable to install the package on target instances.  
  • A JSON-formatted manifest file that describes the package contents. The manifest is not included in the .zip file, but it is stored in the same Amazon S3 bucket as the .zip files that form the package. The manifest identifies the package version and maps the .zip files in the package to target instance attributes, such as operating system version or architecture. For information about how to create the manifest, see Step 2: Create the JSON package manifest (p. 1008).

When you choose *Simple* package creation in the Distributor console, Distributor generates the installation and uninstallation scripts, file hashes, and the JSON package manifest for you, based on the software executable file name and target platforms and architectures.

**Supported package platforms and architectures**

Distributor supports package distribution to any release version of the following platforms that is supported as a Systems Manager managed instance. A version value must match the exact release version of the operating system AMI that you are targeting. For more information about determining this version, see step 4 of Step 2: Create the JSON package manifest (p. 1008).

<table>
<thead>
<tr>
<th>Platform</th>
<th>Code value in manifest file</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server</td>
<td>windows</td>
<td>x86_64 or 386</td>
</tr>
<tr>
<td>Debian</td>
<td>debian</td>
<td>x86_64 or 386</td>
</tr>
</tbody>
</table>
### Getting started with Distributor

Before you use Distributor to create, manage, and deploy software packages, follow these steps.

**Topics**
- **Getting started with Distributor** (p. 1002)
- **Working with Distributor** (p. 1004)
- **Auditing and logging Distributor activity** (p. 1029)
- **Troubleshooting AWS Systems Manager Distributor** (p. 1029)

## Getting started with Distributor

Before you use Distributor, be sure your environment meets the following requirements.

### Step 1: Complete Distributor prerequisites

Before you use Distributor, be sure your environment meets the following requirements.

### Distributor prerequisites

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM Agent</td>
<td>SSM Agent version 2.3.274.0 or later must be installed on the instances to which you want</td>
</tr>
</tbody>
</table>
### Requirement | Description
--- | ---
 | to deploy or from which you want to remove packages.
 | To install or update SSM Agent, see Working with SSM Agent (p. 63).
AWS CLI | (Optional) To use the AWS CLI instead of the AWS Systems Manager console to create and manage packages, install the newest release of the AWS CLI on your local computer.
 | For more information about how to install or upgrade the CLI, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide.
AWS Tools for PowerShell | (Optional) To use the Tools for PowerShell instead of the AWS Systems Manager console to create and manage packages, install the newest release of Tools for PowerShell on your local computer.
 | For more information about how to install or upgrade the Tools for PowerShell, see Setting Up the AWS Tools for Windows PowerShell or AWS Tools for PowerShell Core in the AWS Tools for PowerShell User Guide.

**Note**
Systems Manager currently doesn’t support distributing packages to Oracle Linux instances by using Distributor.

**Step 2: Verify or create an IAM instance profile with Distributor permissions**

By default, AWS Systems Manager doesn’t have permission to perform actions on your instances. You must grant access by using an IAM instance profile. An instance profile is a container that passes IAM role information to an EC2 instance at launch. This requirement applies to permissions for all AWS Systems Manager capabilities, not just Distributor.

If you already use other Systems Manager capabilities, such as Run Command and State Manager, an instance profile with the required permissions for Distributor is already attached to your instances. The simplest way to ensure that you have permissions to perform Distributor tasks is to attach the AmazonSSMManagedInstanceCore policy to your instance profile. For more information, see Create an IAM instance profile for Systems Manager (p. 30).

**Step 3: Control user access to packages**

Using IAM policies, you can control who can create, deploy, and manage packages. You also control which Run Command and State Manager API actions they can perform on managed instances.

**ARN Format**

User-defined packages are associated with document ARNs and have the following format:

```
```
The following is an example.

```
```

You can use a pair of AWS-supplied default IAM policies, one for end users and one for administrators, to grant permissions for Distributor activities. Or you can create custom IAM policies appropriate for your permissions requirements.

For more information about using variables in IAM policies, see IAM Policy Elements: Variables.

For information about how to create policies and attach them to IAM users or groups, see Creating IAM Policies and Adding and Removing IAM Policies in the *IAM User Guide*.

**Step 4: Create or choose an Amazon S3 bucket**

When you create a package by using the Simple workflow in the console, you choose an existing S3 bucket to which Distributor uploads your software. In the Advanced workflow, you must upload .zip files of your software or assets to an S3 bucket before you begin. Whether you create a package by using the Simple or Advanced workflows in the console, or by using the API, you must have an S3 bucket before you start creating your package. As part of the package creation process, Distributor copies your installable software and assets from this bucket to an internal AWS Systems Manager store. Because the assets are copied to an internal store, you can delete or repurpose your S3 bucket when package creation is finished.

For more information about how to create a bucket, see Create a Bucket in the *Amazon Simple Storage Service Getting Started Guide*. For more information about how to run an AWS CLI command to create a bucket, see `mb` in the *AWS CLI Command Reference*.

**Working with Distributor**

You can use the AWS Systems Manager console, AWS command line tools (AWS CLI and AWS Tools for PowerShell), and AWS SDKs to add, manage, or deploy packages in Distributor. Before you add a package to Distributor:

- Create and zip installable assets.
- (Optional) Create a JSON manifest file for the package. This is not required to use the Simple package creation process in the Distributor console. Simple package creation generates a JSON manifest file for you.

You can use the AWS Systems Manager console or a text or JSON editor to create the manifest file.

- Have an S3 bucket ready to store your installable assets or software. If you are using the Advanced package creation process, upload your assets to the S3 bucket before you begin.

  **Note**

  You can delete or repurpose this bucket after you finish creating your package because Distributor moves the package contents to an internal Systems Manager bucket as part of the package creation process.

AWS-published packages are already packaged and ready for deployment. To deploy an AWS-published package to managed instances, see Install or update packages (p. 1019).

**Topics**

- Create a package (p. 1005)
- Edit package permissions (console) (p. 1015)
- Edit package tags (console) (p. 1015)
Create a package

To create a package, prepare your installable software or assets, one file per operating system platform. At least one file is required to create a package.

Different platforms might sometimes use the same file, but all files that you attach to your package must be listed in the *Files* section of the manifest. If you are creating a package by using the simple workflow in the console, the manifest is generated for you. The maximum number of files that you can attach to a single document is 20. The maximum size of each file is 1 GB. For more information about supported platforms, see Supported package platforms and architectures (p. 1001).

When you create a package, you are adding a new SSM document (p. 1031). The document lets you deploy the package to managed instances.

For demonstration purposes only, an example package, `ExamplePackage.zip`, is available for you to download from our website. The example package includes a completed JSON manifest and three .zip files containing installers for PowerShell v7.0.0. The installation and uninstallation scripts do not contain valid commands. Although you must zip each software installable and scripts into a .zip file to create a package in the Advanced workflow, you do not zip installable assets in the Simple workflow.

Create a package (simple)

This section describes how to create a package in Distributor by choosing the Simple package creation workflow in the Distributor console. To create a package, prepare your installable assets, one file per operating system platform. At least one file is required to create a package. The Simple package creation process generates installation and uninstallation scripts, file hashes, and a JSON-formatted manifest for you. The Simple workflow handles the process of uploading and zipping your installable files, and creating a new package and associated SSM document (p. 1031). For more information about supported platforms, see Supported package platforms and architectures (p. 1001).

When you use the Simple method to create a package, Distributor creates install and uninstall scripts for you. However, when you create a package for an in-place update, you must provide your own update script content on the Update script tab. When you add input commands for an update script, Distributor includes this script in the .zip package it creates for you, along with the install and uninstall scripts.

**Note**

The In-place update option lets you add new or updated files to an existing package installation without taking the associated application offline.

To create a package (simple)

2. In the navigation pane, choose Distributor.
3. On the Distributor home page, choose Create package, and then choose Simple.
4. On the Create package page, enter a name for your package. Package names can contain letters, numbers, periods, dashes, and underscores. The name should be generic enough to apply to all versions of the package attachments, but specific enough to identify the purpose of the package.

5. (Optional) For Version name, enter a version name. Version names can be a maximum of 512 characters, and cannot contain special characters.

6. For Location, choose a bucket by using the bucket name and prefix or by using the bucket URL.

7. For Upload software, choose Add software, and then browse for installable software files with .rpm, .msi, or .deb extensions. You can upload more than one software file in a single action.

8. For Target platform, verify that the target operating system platform shown for each installable file is correct. If the operating system shown is not correct, choose the correct operating system from the drop-down list.

For the Simple package creation workflow, because you upload each installable file only once, extra steps are required to instruct Distributor to target a single file at multiple operating systems. For example, if you upload an installable software file named Logtool_v1.1.1.rpm, you must change some defaults in the Simple workflow to target the same software at both Amazon Linux and Ubuntu operating systems. When targeting multiple platforms, do one of the following.

- Use the Advanced workflow instead, zip each installable file into a .zip file before you begin, and manually author the manifest so that one installable file can be targeted at multiple operating system platforms or versions. For more information, see Create a package (advanced) (p. 1007).
- Manually edit the manifest file in the Simple workflow so that your .zip file is targeted at multiple operating system platforms or versions. For more information about how to do this, see the end of step 4 in Step 2: Create the JSON package manifest (p. 1008).

9. For Platform version, verify that the operating system platform version shown is either _any, or the exact, specific operating system release version to which you want your software to apply. For more information about specifying an operating system platform version, see step 4 in Step 2: Create the JSON package manifest (p. 1008).

10. For Architecture, choose the correct processor architecture for each installable file from the drop-down list. For more information about supported processor architectures, see Supported package platforms and architectures (p. 1001).

11. (Optional) Expand Scripts, and review the scripts that Distributor generates for your installable software.

12. (Optional) To provide an update script for use with in-place updates, expand Scripts, choose the Update script tab, and enter your update script commands.

    Systems Manager doesn't generate update scripts on your behalf.

13. To add more installable software files, choose Add software. Otherwise, go to the next step.

14. (Optional) Expand Manifest, and review the JSON package manifest that Distributor generates for your installable software. If you changed any information about your software since you began this procedure, such as platform version or target platform, choose Generate manifest to show the updated package manifest.

    You can edit the manifest manually if you want to target a software installable at more than one operating system, as described in step 8. For more information about editing the manifest, see Step 2: Create the JSON package manifest (p. 1008).

15. Choose Create package.

Wait for Distributor to finish uploading your software and creating your package. Distributor shows upload status for each installable file. Depending on the number and size of packages you are adding, this can take a few minutes. Distributor automatically redirects you to the Package details page for the new package, but you can choose to open this page yourself after the software is uploaded. The Package details page does not show all information about your package until Distributor finishes the package creation process. To stop the upload and package creation process, choose Cancel.
If Distributor cannot upload any of the software installable files, it displays an Upload failed message. To retry the upload, choose Retry upload. For more information about how to troubleshoot package creation failures, see Troubleshooting AWS Systems Manager Distributor (p. 1029).

Create a package (advanced)

In this section, learn about how advanced users can create a package in Distributor after uploading installable assets zipped with installation and uninstallation scripts, and a JSON manifest file, to an S3 bucket.

To create a package, prepare your .zip files of installable assets, one .zip file per operating system platform. At least one .zip file is required to create a package. Next, create a JSON manifest. The manifest includes pointers to your package code files. When you have your required code files added to a folder, and the manifest is populated with correct values, upload your package to an S3 bucket.

An example package, ExamplePackage.zip, is available for you to download from our website. The example package includes a completed JSON manifest and three .zip files.

Topics

- Step 1: Create the ZIP files (p. 1007)
- Step 2: Create the JSON package manifest (p. 1008)
- Step 3: Upload the package and manifest to an S3 bucket (p. 1013)
- Step 4: Add a package to Distributor (p. 1013)

Step 1: Create the ZIP files

The foundation of your package is at least one .zip file of software or installable assets. A package includes one .zip file per operating system that you want to support, unless one .zip file can be installed on multiple operating systems. For example, Red Hat Enterprise Linux and Amazon Linux instances can typically run the same .RPM executable files, so you need to attach only one .zip file to your package to support both operating systems.

Required files

The following items are required in each .zip file:

- An install and an uninstall script. Windows Server-based instances require PowerShell scripts (scripts named install.ps1 and uninstall.ps1). Linux-based instances require shell scripts (scripts named install.sh and uninstall.sh). SSM Agent runs the instructions in the install and uninstall scripts.

  For example, your installation scripts might run an installer (such as .rpm or .msi), they might copy files, or they might set configurations.

- An executable file, installer packages (.rpm, .deb, .msi, etc.), other scripts, or configuration files.

Optional files

The following item is optional in each .zip file:

- An update script. Providing an update script makes it possible for you to use the In-place update option to install a package. When you want to add new or updated files to an existing package installation, the In-place update option does not take the package application offline while the update is performed. Windows Server-based instances require a PowerShell script (script named update.ps1). Linux-based instances require a shell script (script named update.sh). SSM Agent runs the instructions in the update script.

  For more information about installing or updating packages, see Install or update packages (p. 1019).
For examples of .zip files, including sample install and uninstall scripts, download the example package, ExamplePackage.zip.

Step 2: Create the JSON package manifest

After you prepare and zip your installable files, create a JSON manifest. The following is a template. The parts of the manifest template are described in the procedure in this section. You can use a JSON editor to create this manifest in a separate file. Alternatively, you can author the manifest in the AWS Systems Manager console when you create a package.

```json
{
    "schemaVersion": "2.0",
    "version": "your-version",
    "publisher": "optional-publisher-name",
    "packages": {
        "platform": {
            "platform-version": {
                "architecture": {
                    "file": ".zip-file-name-1.zip"
                }
            }
        },
        "another-platform": {
            "platform-version": {
                "architecture": {
                    "file": ".zip-file-name-2.zip"
                }
            }
        },
        "another-platform": {
            "platform-version": {
                "architecture": {
                    "file": ".zip-file-name-3.zip"
                }
            }
        }
    },
    "files": {
        ".zip-file-name-1.zip": {
            "checksums": {
                "sha256": "checksum"
            }
        },
        ".zip-file-name-2.zip": {
            "checksums": {
                "sha256": "checksum"
            }
        }
    }
}
```

To create a JSON package manifest

1. Add the schema version to your manifest. In this release, the schema version is always 2.0.

   ```json
   { "schemaVersion": "2.0",
   }
   ```

2. Add a user-defined package version to your manifest. This is also the value of Version name that you specify when you add your package to Distributor. It becomes part of the AWS Systems Manager document that Distributor creates when you add your package. You also provide this value as an input in the AWS-ConfigureAWSPackage document to install a version of the package other than the latest. A version value can contain letters, numbers, underscores, hyphens, and periods, and

1008
be a maximum of 128 characters in length. We recommend that you use a human-readable package version to make it easier for you and other administrators to specify exact package versions when you deploy. The following is an example.

"version": "1.0.1",

3. (Optional) Add a publisher name. The following is an example.

"publisher": "MyOrganization",

4. Add packages. The "packages" section describes the platforms, release versions, and architectures supported by the .zip files in your package. For more information, see Supported package platforms and architectures (p. 1001).

The platform-version can be the wildcard value, _any. Use it to indicate that a .zip file supports any release of the platform. However, a platform-version value must match the exact release version of the operating system AMI that you are targeting. The following are suggested resources for getting the correct value of the operating system.

- On a Windows Server-based instance, the release version is available as Windows Management Instrumentation (WMI) data. You can run the following Command Prompt command on a Windows Server-based instance to get version information, then parse the results for version. This command does not show the version for Windows Server Nano; the version value for Windows Server Nano is nano.

```
wmic OS get /format:list
```

- On a Linux-based instance, get the version by first scanning for operating system release (the following command). Look for the value of VERSION_ID.

```
cat /etc/os-release
```

If that does not return the results that you need, run the following command to get LSB release information from the /etc/lsb-release file, and look for the value of DISTRIB_RELEASE.

```
lsb_release -a
```

If these methods fail, you can usually find the release based on the distribution. For example, on Debian, you can scan the /etc/debian_version file, or on Red Hat Enterprise Linux, the /etc/redhat-release file.

```
hostnamectl
```

"packages": {
    "platform": {
        "platform-version": {
            "architecture": {
                "file": "zip-file-name-1.zip"
            }
        }
    },
    "another-platform": {
        "platform-version": {
            "architecture": {
                "file": "zip-file-name-2.zip"
            }
        }
    }
}
The following is an example. In this example, the operating system platform is `amazon`, the supported release version is `2016.09`, the architecture is `x86_64`, and the `.zip` file that supports this platform is `test.zip`.

```json
{
    "amazon": {
        "2016.09": {
            "x86_64": {
                "file": "test.zip"
            }
        }
    }
}
```

You can add the `_any` wildcard value to indicate that the package supports all versions of the parent element. For example, to indicate that the package is supported on any release version of Amazon Linux, your package statement should be similar to the following. You can use the `_any` wildcard at the version or architecture levels to support all versions of a platform, or all architectures in a version, or all versions and all architectures of a platform.

```json
{
    "amazon": {
        "_any": {
            "x86_64": {
                "file": "test.zip"
            }
        }
    },
}
```

The following example adds `_any` to show that the first package, `data1.zip`, is supported for all architectures of Amazon Linux `2016.09`. The second package, `data2.zip`, is supported for all releases of Amazon Linux, but only for instances with `x86_64` architecture. Both the `2016.09` and `_any` versions are entries under `amazon`. There is one platform (Amazon Linux), but different supported versions, architectures, and associated `.zip` files.

```json
{
    "amazon": {
        "2016.09": {
            "_any": {
                "file": "data1.zip"
            }
        },
        "_any": {
            "x86_64": {
                "file": "data2.zip"
            }
        }
    }
}
```
You can refer to a `.zip` file more than once in the "packages" section of the manifest, if the `.zip` file supports more than one platform. For example, if you have a `.zip` file that supports both Red Hat Enterprise Linux and Amazon Linux, you have two entries in the "packages" section that point to the same `.zip` file, as shown in the following example.

```json
{
    "amazon": {
        "2018.03": {
            "x86_64": {
                "file": "test.zip"
            }
        }
    },
    "redhat": {
        "_any": {
            "x86_64": {
                "file": "test.zip"
            }
        }
    }
}
```

5. Add the list of `.zip` files that are part of this package from step 4. Each file entry requires the file name and `sha256` hash value checksum. Checksum values in the manifest must match the `sha256` hash value in the zipped assets to prevent the package installation from failing.

To get the exact checksum from your installables, you can run the following commands. On Linux, run `cat file-name.zip | openssl dgst -sha256`. On Windows, run the `Get-FileHash -Path path-to-.zip-file` cmdlet in PowerShell.

The "files" section of the manifest includes one reference to each of the `.zip` files in your package.

```json
"files": {
    "test-agent-x86.deb.zip": {
        "checksums": {
            "sha256": "EXAMPLE2706223c7616ca9fb28863a233b38e5a23a8c326bb4ae241dcEXAMPLE"
        }
    },
    "test-agent-x86_64.deb.zip": {
        "checksums": {
            "sha256": "EXAMPLE572a74504618c491045f25ee6aae8a66307ea9bff0e9d1052EXAMPLE"
        }
    },
    "test-agent-x86_64.nano.zip": {
        "checksums": {
            "sha256": "EXAMPLE63ccb86e830b63dfeef46995af6b32b3c52ce72241b5e80c995EXAMPLE"
        }
    },
    "test-agent-rhel5-x86.nano.zip": {
        "checksums": {
            "sha256": "EXAMPLE13dfe60aa3219bf117638167e5b4ea0a55467e947a363fff0a51EXAMPLE"
        }
    },
    "test-agent-x86.msi.zip": {
        "checksums": {
            "sha256": "EXAMPLE7c25f8c123a3b23c3a23a8c326bb4ae241dcEXAMPLE"
        }
    }
}
```
6. After you add your package information, save and close the manifest file.

The following is an example of a completed manifest. In this example, you have a .zip file, NewPackage_LINUX.zip, that supports more than one platform, but is referenced in the "files" section only once.

```json
{
    "schemaVersion": "2.0",
    "version": "1.7.1",
    "publisher": "Amazon Web Services",
    "packages": {
        "windows": {
            "_any": {
                "x86_64": {
                    "file": "NewPackage_WINDOWS.zip"
                }
            }
        },
        "amazon": {
            "_any": {
                "x86_64": {
                    "file": "NewPackage_LINUX.zip"
                }
            }
        },
        "ubuntu": {
            "_any": {
                "x86_64": {
                    "file": "NewPackage_LINUX.zip"
                }
            }
        }
    },
    "files": {
        "NewPackage_WINDOWS.zip": {
            "checksums": {
                "sha256": "EXAMPLEc2c706013cf8c68163459678f7f6daa9489cd3f91d52799331EXAMPLE"
            }
        },
        "NewPackage_LINUX.zip": {
            "checksums": {
                "sha256": "EXAMPLE12a4abbd10315a668a7384cc9b5ca8ad8e9ced8ef1bf0e5478EXAMPLE"
            }
        }
    }
}
```
Package example

An example package, ExamplePackage.zip, is available for you to download from our website. The example package includes a completed JSON manifest and three .zip files.

Step 3: Upload the package and manifest to an S3 bucket

Prepare your package by copying or moving all .zip files into a folder or directory. A valid package requires the manifest that you created in Step 2: Create the JSON package manifest (p. 1008) and all .zip files identified in the manifest file list.

To upload the package and manifest to Amazon S3

1. Copy or move all .zip archive files that you specified in the manifest to a folder or directory. Do not zip the folder or directory you move your .zip archive files and manifest file to.
2. Create a bucket or choose an existing bucket. For more information, see Create a Bucket in the Amazon Simple Storage Service Getting Started Guide. For more information about how to run an AWS CLI command to create a bucket, see mb in the AWS CLI Command Reference.
3. Upload the folder to the bucket. For more information, see Add an Object to a Bucket in the Amazon Simple Storage Service Getting Started Guide. If you plan to paste your JSON manifest into the AWS Systems Manager console, do not upload the manifest. For more information about how to run an AWS CLI command to upload files to a bucket, see mv in the AWS CLI Command Reference.
4. On the bucket’s home page, choose the folder that you uploaded. If you uploaded your files to a subfolder in a bucket, be sure to note the subfolder (also known as a prefix). You need the prefix to add your package to Distributor.

Step 4: Add a package to Distributor

You can use the AWS Systems Manager console, AWS command line tools (AWS CLI and AWS Tools for PowerShell), or AWS SDks to add a new package to AWS Systems Manager Distributor. When you add a package, you are adding a new SSM document (p. 1031). The document lets you deploy the package to managed instances.

Topics
- Adding a package (console) (p. 1013)
- Adding a package (AWS CLI) (p. 1014)

Adding a package (console)

You can use the AWS Systems Manager console to create a package. Have ready the name of the bucket to which you uploaded your package in Step 3: Upload the package and manifest to an S3 bucket (p. 1013).

To add a package to Distributor (console)

2. In the navigation pane, choose **Distribution**.
3. On the Distributor home page, choose **Create package**, and then choose **Advanced**.
4. On the **Create package** page, enter a name for your package. Package names can contain letters, numbers, periods, dashes, and underscores. The name should be generic enough to apply to all versions of the package attachments, but specific enough to identify the purpose of the package.
5. For **Version name**, enter the exact value of the version entry in your manifest file.
6. For **S3 bucket name**, choose the name of the bucket to which you uploaded your .zip files and manifest in the section called “Step 3: Upload the package and manifest to an S3 bucket” (p. 1013).
7. For **S3 key prefix**, enter the subfolder of the bucket where your .zip files and manifest are stored.
8. For **Manifest**, choose **Extract from package** to use a manifest that you have uploaded to the S3 bucket with your .zip files. 

(Optional) If you did not upload your JSON manifest to the S3 bucket where you stored your .zip files, choose **New manifest**. You can author or paste the entire manifest in the JSON editor field. For more information about how to create the JSON manifest, see Step 2: Create the JSON package manifest (p. 1008).
9. When you are finished with the manifest, choose **Create package**.
10. Wait for Distributor to create your package from your .zip files and manifest. Depending on the number and size of packages you are adding, this can take a few minutes. Distributor automatically redirects you to the **Package details** page for the new package, but you can choose to open this page yourself after the software is uploaded. The **Package details** page does not show all information about your package until Distributor finishes the package creation process. To stop the upload and package creation process, choose **Cancel**.

**Adding a package (AWS CLI)**

You can use the AWS CLI to create a package. Have the URL ready from the bucket to which you uploaded your package in Step 3: Upload the package and manifest to an S3 bucket (p. 1013).

**To add a package to Amazon S3 (AWS CLI)**

1. To use the AWS CLI to create a package, run the following command, replacing **package-name** with the name of your package and **path-to-manifest-file** with the file path for your JSON manifest file. **S3-bucket-URL-of-package** is the URL of the S3 bucket where the entire package is stored. When you run the **create-document** command in Distributor, you specify the **Package** value for **--document-type**.

If you did not add your manifest file to the S3 bucket, the **--content** parameter value is the file path to the JSON manifest file.

```bash
aws ssm create-document
   --name "package-name"
   --content file://path-to-manifest-file
   --attachments Key="SourceUrl",Values="S3-bucket-URL-of-package"
   --version-name version-value-from-manifest
   --document-type Package
```

The following is an example.

```bash
aws ssm create-document
   --name "ExamplePackage"
   --content file://path-to-manifest-file
   --attachments Key="SourceUrl",Values="https://s3.amazonaws.com/mybucket/ExamplePackage"
   --version-name 1.0.1
```
2. Verify that your package was added and show the package manifest by running the following command, replacing `package-name` with the name of your package. To get a specific version of the document (not the same as the version of a package), you can add the `--document-version` parameter.

```
aws ssm get-document \
  --name "package-name"
```

For information about other options you can use with the `create-document` command, see `create-document` in the AWS Systems Manager section of the AWS CLI Command Reference. For information about other options you can use with the `get-document` command, see `get-document`.

## Edit package permissions (console)

After you have added a package to AWS Systems Manager Distributor, you can edit the package's permissions in the AWS Systems Manager console. You can add other AWS accounts to a package's permissions. By default, packages are set to `Private`, meaning only those with access to the package creator's AWS account can view package information and update or delete the package. If `Private` permissions are acceptable, you can skip this procedure.

### To edit package permissions (console)

2. In the navigation pane, choose `Distributor`.
3. On the Packages page, choose the package for which you want to edit permissions.
4. On the Package details tab, choose `Edit permissions` to change permissions.
5. For `Edit permissions`, choose `Shared with specific accounts`.
6. Under `Shared with specific accounts`, add AWS account numbers, one at a time. When you are finished, choose `Save`.

**Note**

You cannot share packages with all accounts.

## Edit package tags (console)

After you have added a package to AWS Systems Manager Distributor, you can edit the package's tags in the AWS Systems Manager console. These tags are applied to the package, and are not connected to tags on the instances to which you want to deploy the package. Tags are case sensitive key and value pairs that can help you group and filter your packages by criteria that are relevant to your organization. If you do not want to add tags, you are ready to install your package or add a new version.

### To edit package tags (console)

2. In the navigation pane, choose `Distributor`.
3. On the Packages page, choose the package for which you want to edit tags.
5. For `Add tags`, enter a tag key, or a tag key and value pair, and then choose `Add`. Repeat if you want to add more tags. To delete tags, choose `X` on the tag at the bottom of the window.
6. When you are finished adding tags to your package, choose `Save`. 

1015
Add a package version to Distributor

To add a package version, create a package (p. 1005), and then use Distributor to add a package version by adding an entry to the SSM document that already exists for older versions. To save time, update the manifest for an older version of the package, change the value of the version entry in the manifest (for example, from Test_1.0 to Test_2.0) and save it as the manifest for the new version. The simple Add version workflow in the Distributor console updates the manifest file for you.

A new package version can:
- Replace at least one of the installable files attached to the current version.
- Add new installable files to support additional platforms.
- Delete files to discontinue support for specific platforms.

A newer version can use the same S3 bucket, but must have a URL with a different file name shown at the end. You can use the AWS Systems Manager console or the AWS CLI to add the new version. Uploading an installable file with the exact name as an existing installable file in the S3 bucket overwrites the existing file. No installable files are copied over from the older version to the new version; you must upload installable files from the older version to have them be part of a new version. After Distributor is finished creating your new package version, you can delete or repurpose the S3 bucket, because Distributor copies your software to an internal Systems Manager bucket as part of the versioning process.

**Note**
Each package is held to a maximum of 25 versions. You can delete versions that are no longer required.

**Topics**
- Adding a package version (console) (p. 1016)
- Adding a package version (AWS CLI) (p. 1018)

Adding a package version (console)

Before you perform these steps, follow the instructions in Create a package (p. 1005) to create a new package for the version. Then, use the AWS Systems Manager console to add a new package version to Distributor.

Adding a package version (simple)

To add a package version by using the Simple workflow, prepare updated installable files or add installable files to support more platforms and architectures. Then, use Distributor to upload new and updated installable files and add a package version. The simplified Add version workflow in the Distributor console updates the manifest file and associated SSM document for you.

**To add a package version (simple)**

2. In the navigation pane, choose Distributor.
3. On the Distributor home page, choose the package to which you want to add another version.
4. On the Add version page, choose Simple.
5. For Version name, enter a version name. The version name for the new version must be different from older version names. Version names can be a maximum of 512 characters, and cannot contain special characters.
6. **For S3 bucket name**, choose an existing S3 bucket from the list. This can be the same bucket that you used to store installable files for older versions, but the installable file names must be different to avoid overwriting existing installable files in the bucket.

7. **For S3 key prefix**, enter the subfolder of the bucket where your installable assets are stored.

8. **For Upload software**, browse for the installable software files that you want to attach to the new version. Installable files from existing versions are not automatically copied over to a new version; you must upload any installable files from older versions of the package if you want any of the same installable files to be part of the new version. You can upload more than one software file in a single action.

9. **For Target platform**, verify that the target operating system platform shown for each installable file is correct. If the operating system shown is not correct, choose the correct operating system from the drop-down list.

In the Simple versioning workflow, because you upload each installable file only once, extra steps are required to target a single file at multiple operating systems. For example, if you upload an installable software file named `Logtool_v1.1.1.rpm`, you must change some defaults in the Simple workflow to instruct Distributor to target the same software at both Amazon Linux and Ubuntu operating systems. You can do one of the following to work around this limitation.

- Use the Advanced versioning workflow instead, zip each installable file into a .zip file before you begin, and manually author the manifest so that one installable file can be targeted at multiple operating system platforms or versions. For more information, see Adding a package version (advanced) (p. 1018).

- Manually edit the manifest file in the Simple workflow so that your .zip file is targeted at multiple operating system platforms or versions. For more information about how to do this, see the end of step 4 in Step 2: Create the JSON package manifest (p. 1008).

10. For **Platform version**, verify that the operating system platform version shown is either _any_, or the exact, specific operating system release version to which you want your software to apply. For more information about specifying a platform version, see step 4 in Step 2: Create the JSON package manifest (p. 1008).

11. For **Architecture**, choose the correct processor architecture for each installable file from the drop-down list. For more information about supported architectures, see Supported package platforms and architectures (p. 1001).

12. **(Optional) Expand Scripts**, and review the installation and uninstallation scripts that Distributor generates for your installable software.

13. To add more installable software files to the new version, choose **Add software**. Otherwise, go to the next step.

14. **(Optional) Expand Manifest**, and review the JSON package manifest that Distributor generates for your installable software. If you changed any information about your installable software since you began this procedure, such as platform version or target platform, choose **Generate manifest** to show the updated package manifest.

You can edit the manifest manually if you want to target a software installable at more than one operating system, as described in step 9. For more information about editing the manifest, see Step 2: Create the JSON package manifest (p. 1008).

15. When you finish adding software and reviewing the target platform, version, and architecture data, choose **Add version**.

16. **Wait for Distributor to finish uploading your software and creating the new package version.** Distributor shows upload status for each installable file. Depending on the number and size of packages you are adding, this can take a few minutes. Distributor automatically redirects you to the Package details page for the package, but you can choose to open this page yourself after the software is uploaded. The Package details page does not show all information about your package until Distributor finishes creating the new package version. To stop the upload and package version creation, choose **Stop upload**.
17. If Distributor cannot upload any of the software installable files, it displays an Upload failed message. To retry the upload, choose Retry upload. For more information about how to troubleshoot package version creation failures, see Troubleshooting AWS Systems Manager Distributor (p. 1029).

18. When Distributor is finished creating the new package version, on the package’s Details page, on the Versions tab, view the new version in the list of available package versions. Set a default version of the package by choosing a version, and then choosing Set default version.

If you do not set a default version, the newest package version is the default version.

Adding a package version (advanced)

To add a package version, create a package (p. 1005), and then use Distributor to add a package version by adding an entry to the SSM document that exists for older versions. To save time, update the manifest for an older version of the package, change the value of the version entry in the manifest (for example, from Test_1.0 to Test_2.0) and save it as the manifest for the new version. You must have an updated manifest to add a new package version by using the Advanced workflow.

To add a package version (advanced)

2. In the navigation pane, choose Distributor.
3. On the Distributor home page, choose the package to which you want to add another version, and then choose Add version.
4. For Version name, enter the exact value that is in the version entry of your manifest file.
5. For S3 bucket name, choose an existing S3 bucket from the list. This can be the same bucket that you used to store installable files for older versions, but the installable file names must be different to avoid overwriting existing installable files in the bucket.
6. For S3 key prefix, enter the subfolder of the bucket where your installable assets are stored.
7. For Manifest, choose Extract from package to use a manifest that you uploaded to the S3 bucket with your .zip files.
   
   (Optional) If you did not upload your revised JSON manifest to the S3 bucket where you stored your .zip files, choose New manifest. You can author or paste the entire manifest in the JSON editor field. For more information about how to create the JSON manifest, see Step 2: Create the JSON package manifest (p. 1008).
8. When you are finished with the manifest, choose Add package version.
9. On the package’s Details page, on the Versions tab, view the new version in the list of available package versions. Set a default version of the package by choosing a version, and then choosing Set default version.

If you do not set a default version, the newest package version is the default version.

Adding a package version (AWS CLI)

You can use the AWS CLI to add a new package version to Distributor. Before you run these commands, you must create a new package version and upload it to S3, as described at the start of this topic.

To add a package version (AWS CLI)

1. Run the following command to edit the AWS Systems Manager document with an entry for a new package version. Replace document-name with the name of your document. Replace S3-bucket-URL-to-manifest-file with the URL of the JSON manifest that you copied in Step 3: Upload the package and manifest to an S3 bucket (p. 1013). S3-bucket-URL-of-package is the URL of the S3 bucket where the entire package is stored. Replace version-name-from-updated-manifest with the version name as defined in the updated manifest file.
with the value of `version` in the manifest. Set the `--document-version` parameter to `$LATEST` to make the document associated with this package version the latest version of the document.

```bash
```

The following is an example.

```bash
```

2. Run the following command to verify that your package was updated and show the package manifest. Replace `package-name` with the name of your package, and optionally, `document-version` with the version number of the document (not the same as the package version) that you updated. If this package version is associated with the latest version of the document, you can specify `$LATEST` for the value of the optional `--document-version` parameter.

```bash
aws ssm get-document --name "package-name" --document-version "document-version"
```

For information about other options you can use with the `update-document` command, see `update-document` in the AWS Systems Manager section of the AWS CLI Command Reference.

## Install or update packages

You can use the AWS Management Console or the AWS CLI to deploy packages to your AWS Systems Manager managed instances by using AWS Systems Manager Distributor. You can currently deploy one version of one package per command. You can install new packages or update existing installations in place. You can choose to deploy a specific version or choose to always deploy the latest version of a package for deployment.

<table>
<thead>
<tr>
<th>Preference</th>
<th>AWS Systems Manager action</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install or update a package immediately.</td>
<td>Run Command</td>
<td>• Installing or updating a package one time (console) (p. 1020)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Installing a package one time (AWS CLI) (p. 1022)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updating a package one time (AWS CLI) (p. 1023)</td>
</tr>
<tr>
<td>Install or update a package on a schedule, so that</td>
<td>State Manager</td>
<td>• Scheduling a package installation or update (console) (p. 1021)</td>
</tr>
<tr>
<td>the installation always includes the default version.</td>
<td></td>
<td>• Scheduling a package installation (AWS CLI) (p. 1023)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Scheduling a package update (AWS CLI) (p. 1024)</td>
</tr>
<tr>
<td>Automatically install a package on new instances that</td>
<td>State Manager</td>
<td>One way to do this is to apply tags to new instances, and then</td>
</tr>
<tr>
<td>have a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Installing or updating a package one time (console)

You can use the AWS Systems Manager console to install or update a package one time. When you configure a one-time installation, Distributor uses AWS Systems Manager Run Command (p. 829) to perform the installation.

To install or update a package one time (console)

2. In the navigation pane, choose Distributor.
3. On the Distributor home page, choose the package that you want to install.
4. Choose Install one time.

   This command opens Systems Manager Run Command with the command document AWS-ConfigureAWSPackage and your Distributor package already selected.
5. For Document version, select the version of the AWS-ConfigureAWSPackage document that you want to run.
6. For Action, choose Install.
7. For Installation type, choose one of the following:
   - Uninstall and reinstall: The package is completely uninstalled, and then reinstalled. The application is unavailable until the reinstallation completes.
   - In-place update: Only new or changed files are added to the existing installation according your instructions you provide in an update script. The application remains available throughout the update process
8. For Name, verify that the name of the package you selected is entered.
9. (Optional) For Version, enter the version name value of the package. If you leave this field blank, Run Command installs the default version that you selected in Distributor.
10. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

   **Note**
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.
11. For **Other parameters:**
   - For **Comment**, type information about this command.
   - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

12. (Optional) For **Rate control**:
   - For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
     
     **Note**
     If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
   - For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

13. (Optional) For **Output options**, to save the command output to a file, select the Write command output to an S3 bucket box. Type the bucket and prefix (folder) names in the boxes.

     **Note**
     The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

14. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the Enable SNS notifications check box.

     For more information about configuring Amazon SNS notifications for Run Command, see Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

15. When you are ready to install the package, choose **Run**.

16. The **Command status** area reports the progress of the execution. If the command is still in progress, choose the refresh icon in the top-left corner of the console until the **Overall status** or **Detailed status** column shows **Success** or **Failed**.

17. In the **Targets and outputs** area, choose the button next to an instance name, and then choose **View output**.

     The command output page shows the results of your command execution.

18. (Optional) If you chose to write command output to an S3 bucket, choose **Amazon S3** to view the output log data.

**Scheduling a package installation or update (console)**

You can use the AWS Systems Manager console to schedule the installation or update of a package. When you schedule package installation or update, Distributor uses AWS Systems Manager State Manager (p. 864) to install or update.

**To schedule a package installation (console)**

2. In the navigation pane, choose **Distributor**.
3. On the Distributor home page, choose the package that you want to install or update.
4. For Package, choose Install on a schedule.

This command opens Systems Manager State Manager to a new association that is created for you.

5. For Name, enter a name (for example, Deploy-test-agent-package). This is optional, but recommended. Spaces aren't allowed in the name.

6. In the Document list, the document name AWS-ConfigureAWSPackage is already selected.

7. For Action, verify that Install is selected.

8. For Installation type, choose one of the following:

   - Uninstall and reinstall: The package is completely uninstalled, and then reinstalled. The application is unavailable until the reinstallation completes.
   - In-place update: Only new or changed files are added to the existing installation according to instructions you provide in an update script. The application remains available throughout the update process.

9. For Name, verify that the name of your package is entered.

10. For Version, if you want to install a package version other than the latest published version, enter the version identifier.

11. For Targets, choose Selecting all managed instances in this account, Specifying tags, or Manually Selecting Instance. If you target resources by using tags, enter a tag key and a tag value in the fields provided.

12. For Specify schedule, choose On Schedule to run the association on a regular schedule, or No Schedule to run the association once. For more information about these options, see Create an association (p. 870). Use the controls to create a cron or rate schedule for the association.

13. Choose Create Association.

14. On the Association page, choose the button next to the association you created, and then choose Apply association now.

   State Manager creates and immediately runs the association on the specified instances or targets. For more information about the results of running associations, see Create an association (p. 870) in this guide.

For more information about working with the options in Advanced options, Rate control, and Output options, see Create an association (p. 870).

Installing a package one time (AWS CLI)

You can run send-command in the AWS CLI to install a Distributor package one time. If the package is already installed, the application will be taken offline while the package is uninstalled and the new version installed in its place.

To install a package one time (AWS CLI)

- Run the following command in the AWS CLI.

  ```bash
  aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "instance-IDs" --parameters '{"action": ["Install"], "installationType": ["Uninstall and reinstall"], "name": ["package-name (in same account) or package-ARN (shared from different account)"]}'
  ```

Note

The default behavior for installationType is Uninstall and reinstall. You can omit "installationType": ["Uninstall and reinstall"] from this command when you are installing a complete package.
The following is an example.

```bash
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "i-00000000000000" --parameters '{"action": "Install","installationType": ["Uninstall and reinstall"], "name": "ExamplePackage"]'
```

For information about other options you can use with the `ssm send-command` command, see `send-command` in the AWS Systems Manager section of the AWS CLI Command Reference.

**Updating a package one time (AWS CLI)**

You can run `ssm send-command` in the AWS CLI to update a Distributor package without taking the associated application offline. Only new or updated files in the package are replaced.

**To update a package one time (AWS CLI)**

- Run the following command in the AWS CLI.

```bash
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "instance-IDs" --parameters '{"action": ["Install"],"installationType": ["In-place update"],"name": ["package-name (in same account) or package-ARN (shared from different account)"]}'
```

**Note**
When you add new or changed files, you must include "installationType": ["In-place update"] in the command.

The following is an example.

```bash
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "i-02573cafcfEXAMPLE" --parameters '{"action": ["Install"],"installationType": ["In-place update"],"name": ["ExamplePackage"]}'
```

For information about other options you can use with the `ssm send-command` command, see `send-command` in the AWS Systems Manager section of the AWS CLI Command Reference.

**Scheduling a package installation (AWS CLI)**

You can run `create-association` in the AWS CLI to install a Distributor package on a schedule. The value of `--name`, the document name, is always AWS-ConfigureAWSPackage. The following command uses the key `InstanceIds` to specify target instances. If the package is already installed, the application will be taken offline while the package is uninstalled and the new version installed in its place.

```bash
aws ssm create-association --name "AWS-ConfigureAWSPackage" --parameters '{"action": ["Install"],"installationType": ["Uninstall and reinstall"],"name": ["package-name (in same account) or package-ARN (shared from different account)"]}' --targets [{"Key": "InstanceIds", "Values": ["instance-ID1", "instance-ID2"]}]
```

**Note**
The default behavior for `installationType` is Uninstall and reinstall. You can omit "installationType": ["Uninstall and reinstall"] from this command when you are installing a complete package.

The following is an example.
aws ssm create-association --name "AWS-ConfigureAWSPackage" --parameters '{"action": ["Install"], "installationType": ["Install and reinstall"], "name": ["Test-ConfigureAWSPackage"], "targets": [{"Key": "InstanceIds", "Values": ["i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE"]}]}

For information about other options you can use with the `create-association` command, see `create-association` in the AWS Systems Manager section of the AWS CLI Command Reference.

**Scheduling a package update (AWS CLI)**

You can run `create-association` in the AWS CLI to update a Distributor package on a schedule without taking the associated application offline. Only new or updated files in the package are replaced. The value of `--name`, the document name, is always AWS-ConfigureAWSPackage. The following command uses the key `InstanceIds` to specify target instances.

```bash
aws ssm create-association --name "AWS-ConfigureAWSPackage" --parameters '{"action": ["Install"], "installationType": ["In-place update"], "name": ["package-name (in same account) or package-ARN (shared from different account)"], "targets": [{"Key": "InstanceIds", "Values": ["instance-ID1", "instance-ID2"]}]}
```

**Note**

When you add new or changed files, you must include "installationType": ["In-place update"] in the command.

The following is an example.

```bash
aws ssm create-association --name "AWS-ConfigureAWSPackage" --parameters '{"action": ["Install"], "installationType": ["In-place update"], "name": ["Test-ConfigureAWSPackage"], "targets": [{"Key": "InstanceIds", "Values": ["i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE"]}]}
```

For information about other options you can use with the `create-association` command, see `create-association` in the AWS Systems Manager section of the AWS CLI Command Reference.

**Uninstall a package**

You can use the AWS Management Console or the AWS CLI to uninstall Distributor packages from your AWS Systems Manager managed instances by using Run Command. In this release, you can uninstall one version of one package per command. You can uninstall a specific version or the default version.

**Topics**

- Uninstalling a package (console) (p. 1024)
- Uninstalling a package (AWS CLI) (p. 1025)

**Uninstalling a package (console)**

You can use Run Command in the AWS Systems Manager console to uninstall a package one time. Distributor uses AWS Systems Manager Run Command (p. 829) to uninstall packages.

**To uninstall a package (console)**

2. In the navigation pane, choose Run Command.
5. From Action, choose Uninstall
6. For Name, enter the name of the package that you want to uninstall.
7. In Targets, choose target managed instances by specifying a tag key and values that are shared by the managed instances, or specify instances by any of seven attributes, including instance ID, platform, and SSM Agent version.
8. You can use the advanced options to add comments about the operation, change Concurrency and Error threshold values in Rate control, specify output options, or configure Amazon SNS notifications. For more information, see Running Commands from the Console in this guide.
9. When you are ready to uninstall the package, choose Run, and then choose View results.
10. In the commands list, choose the AWS-ConfigureAWSPackage command that you ran. If the command is still in progress, choose the refresh icon in the top-right corner of the console.
11. When the Status column shows Success or Failed, choose the Output tab.
12. Choose View output. The command output page shows the results of your command execution.

Uninstalling a package (AWS CLI)

You can use the AWS CLI to uninstall a Distributor package from managed instances by using Run Command.

To uninstall a package (AWS CLI)

- Run the following command in the AWS CLI.

```bash
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "instance-IDs" --parameters '{"action":"Uninstall","name":"package-name (in same account) or package-ARN (shared from different account)"}'
```

The following is an example.

```bash
aws ssm send-command --document-name "AWS-ConfigureAWSPackage" --instance-ids "i-02573cafcfEXAMPLE" --parameters '{"action":"Uninstall","name":"Test-ConfigureAWSPackage"}'
```

For information about other options you can use with the send-command command, see send-command in the AWS Systems Manager section of the AWS CLI Command Reference.

Delete a package

Deleting a package (console)

You can use the AWS Systems Manager console to delete a package from Distributor. Deleting a package deletes all versions of a package from Distributor.

To delete a package (console)

2. In the navigation pane, choose Distributor.
3. On the Distributor home page, choose the package that you want to delete.
4. On the package's details page, choose **Delete package**.
5. When you are prompted to confirm the deletion, choose **Delete package**.

**Deleting a package version (console)**

You can use the AWS Systems Manager console to delete a package version from Distributor.

**To delete a package version (console)**

2. In the navigation pane, choose **Distributor**.
3. On the **Distributor** home page, choose the package that you want to delete a version of.
4. On the versions page for the package, choose the version to delete and choose **Delete version**.
5. When you are prompted to confirm the deletion, choose **Delete package version**.

**Deleting a package (command line)**

You can use your preferred command line tool to delete a package from Distributor.

**Linux**

**To delete a package (AWS CLI)**

1. Run the following command to list documents for specific packages. In the results of this command, look for the package that you want to delete.

   ```bash
   aws ssm list-documents \
   --filters Key=Name,Values=package-name
   ```

2. Run the following command to delete a package. Replace `package-name` with the package name.

   ```bash
   aws ssm delete-document \
   --name "package-name"
   ```

3. Run the `list-documents` command again to verify that the package was deleted. The package you deleted should not appear in the list.

   ```bash
   aws ssm list-documents \
   --filters Key=Name,Values=package-name
   ```

**Windows**

**To delete a package (AWS CLI)**

1. Run the following command to list documents for specific packages. In the results of this command, look for the package that you want to delete.

   ```bash
   aws ssm list-documents ^
   --filters Key=Name,Values=package-name
   ```

2. Run the following command to delete a package. Replace `package-name` with the package name.

   ```bash
   aws ssm delete-document ^
   --name "package-name"
   ```

1026
aws ssm delete-document 
   --name "package-name"

3. Run the `list-documents` command again to verify that the package was deleted. The package you deleted should not appear in the list.

aws ssm list-documents 
   --filters Key=Name,Values=package-name

**PowerShell**

**To delete a package (Tools for PowerShell)**

1. Run the following command to list documents for specific packages. In the results of this command, look for the package that you want to delete.

   $filter.Key = "Name"
   $filter.Values = "package-name"

   Get-SSMDocumentList 
   -Filters @($filter)

2. Run the following command to delete a package. Replace `package-name` with the package name.

   Remove-SSMDocument 
   -Name "package-name"

3. Run the `Get-SSMDocumentList` command again to verify that the package was deleted. The package you deleted should not appear in the list.

   $filter.Key = "Name"
   $filter.Values = "package-name"

   Get-SSMDocumentList 
   -Filters @($filter)

**Deleting a package version (command line)**

You can use your preferred command line tool to delete a package version from Distributor.

**Linux**

**To delete a package version (AWS CLI)**

1. Run the following command to list the versions of your package. In the results of this command, look for the package version that you want to delete.

   aws ssm list-document-versions 
   --name "package-name"

2. Run the following command to delete a package version. Replace `package-name` with the package name and `version` with the version number.
aws ssm delete-document
   --name "package-name"
   --document-version version

3. Run the `list-document-versions` command to verify that the version of the package was deleted. The package version that you deleted should not be found.

aws ssm list-document-versions
   --name "package-name"

Windows

To delete a package version (AWS CLI)

1. Run the following command to list the versions of your package. In the results of this command, look for the package version that you want to delete.

aws ssm list-document-versions
   --name "package-name"

2. Run the following command to delete a package version. Replace `package-name` with the package name and `version` with the version number.

aws ssm delete-document
   --name "package-name"
   --document-version version

3. Run the `list-document-versions` command to verify that the version of the package was deleted. The package version that you deleted should not be found.

aws ssm list-document-versions
   --name "package-name"

PowerShell

To delete a package version (Tools for PowerShell)

1. Run the following command to list the versions of your package. In the results of this command, look for the package version that you want to delete.

`Get-SSMDocumentVersionList` `\`
   -Name "package-name"
\`

2. Run the following command to delete a package version. Replace `package-name` with the package name and `version` with the version number.

`Remove-SSMDocument` `\`
   -Name "package-name"
   -DocumentVersion version
\`

3. Run the `Get-SSMDocumentVersionList` command to verify that the version of the package was deleted. The package version that you deleted should not be found.

`Get-SSMDocumentVersionList` `\`
For information about other options you can use with the `list-documents` command, see `list-documents` in the AWS Systems Manager section of the AWS CLI Command Reference. For information about other options you can use with the `delete-document` command, see `delete-document`.

Auditing and logging Distributor activity

For more information about auditing and logging options for AWS Systems Manager, see Monitoring AWS Systems Manager (p. 1134).

Audit Distributor activity using AWS CloudTrail

AWS CloudTrail captures API calls made in the Systems Manager console, the AWS CLI, and the Systems Manager SDK. The information can be viewed in the CloudTrail console or stored in an S3 bucket. One bucket is used for all CloudTrail logs for your account.

Logs of Run Command and State Manager actions show document creation, package installation, and package uninstallation activity. For more information about viewing and using CloudTrail logs of Systems Manager activity, see Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144).

Troubleshooting AWS Systems Manager Distributor

The following information can help you troubleshoot problems that might occur when you use Distributor.

Topics

- Wrong package with the same name is installed (p. 1029)
- Error: Failed to retrieve manifest: Could not find latest version of package (p. 1029)
- Error: Failed to retrieve manifest: Validation exception (p. 1030)

Wrong package with the same name is installed

Problem: You've installed a package, but AWS Systems Manager Distributor installed a different package instead.

Cause: During installation, AWS Systems Manager finds AWS-published packages as results before user-defined external packages. If your user-defined package name is the same as an AWS-published package name, the AWS package is installed instead of your package.

Solution: To avoid this problem, name your package something different from the name for an AWS-published package.

Error: Failed to retrieve manifest: Could not find latest version of package

Problem: You received an error like the following:

```
Failed to retrieve manifest: ResourceNotFoundException: Could not find the latest version of package
arn:aws:ssm:::package/package-name status code: 400, request id: guid
```
Cause: You are using a version of SSM Agent with Systems Manager Distributor that is earlier than version 2.3.274.0.

Solution: Update the version of SSM Agent to version 2.3.274.0 or later. For more information, see Update SSM Agent by using Run Command (p. 835) or Automatically update SSM Agent (CLI) (p. 908).

Error: Failed to retrieve manifest: Validation exception

Problem: You received an error like the following:

```
Failed to retrieve manifest: ValidationException: 1 validation error detected: Value 'documentArn' at 'packageName' failed to satisfy constraint: Member must satisfy regular expression pattern: arn:aws:ssm:region-id:account-id:package/package-name
```

Cause: You are using a version of SSM Agent with Systems Manager Distributor that is earlier than version 2.3.274.0.

Solution: Update the version of SSM Agent to version 2.3.274.0 or later. For more information, see Update SSM Agent by using Run Command (p. 835) or Automatically update SSM Agent (CLI) (p. 908).
AWS Systems Manager Shared Resources

Systems Manager uses the following shared resources for managing and configuring your AWS resources.

Topics
• AWS Systems Manager documents (p. 1031)

AWS Systems Manager documents

An AWS Systems Manager document (SSM document) defines the actions that Systems Manager performs on your managed instances. Systems Manager includes more than a dozen pre-configured documents that you can use by specifying parameters at runtime. Documents use JavaScript Object Notation (JSON) or YAML, and they include steps and parameters that you specify.

Types of SSM documents

The following table describes the different types of SSM documents.

<table>
<thead>
<tr>
<th>Type</th>
<th>Use with</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command document</td>
<td>Run Command (p. 829)</td>
<td>Run Command uses command documents to run commands. State Manager uses command documents to apply a configuration. These actions can be run on one or more targets at any point during the lifecycle of an instance. Maintenance Windows uses command documents to apply a configuration based on the specified schedule.</td>
</tr>
<tr>
<td></td>
<td>State Manager (p. 864)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintenance Windows (p. 620)</td>
<td></td>
</tr>
<tr>
<td>Automation document</td>
<td>Automation (p. 278)</td>
<td>Use automation documents when performing common maintenance and deployment tasks such as creating or updating an Amazon Machine Image (AMI). State Manager uses automation documents to apply a configuration. These actions can be run on one or more targets at any point during the lifecycle of an instance. Maintenance Windows uses automation documents to perform common maintenance.</td>
</tr>
<tr>
<td></td>
<td>State Manager (p. 864)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintenance Windows (p. 620)</td>
<td></td>
</tr>
</tbody>
</table>
### SSM documents

<table>
<thead>
<tr>
<th>Type</th>
<th>Use with</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package document</td>
<td>Distributor (p. 999)</td>
<td>In Distributor, a package is represented by an SSM document. A package document includes attached ZIP archive files that contain software or assets to install on managed instances. Creating a package in Distributor creates the package document.</td>
</tr>
<tr>
<td>Session document</td>
<td>Session Manager (p. 770)</td>
<td>Session Manager uses session documents to determine which type of session to start, such as a port forwarding session, a session to run an interactive command, or a session to create an SSH tunnel.</td>
</tr>
<tr>
<td>Policy document</td>
<td>State Manager (p. 864)</td>
<td>Systems Manager Inventory uses the AWS-GatherSoftwareInventory policy document with a State Manager association to collect inventory data from managed instances. When creating your own SSM documents, Automation documents and Run Command documents are the preferred method for enforcing a policy on a managed instance.</td>
</tr>
<tr>
<td>Change Calendar document</td>
<td>Change Calendar (p. 613)</td>
<td>Systems Manager Change Calendar uses the ChangeCalendar document type. A Change Calendar document stores a calendar entry and associated events that can allow or prevent Automation actions from changing your environment. In Change Calendar, a document stores iCalendar 2.0 data in plain-text format.</td>
</tr>
</tbody>
</table>

### SSM document versions and execution

You can create and save different versions of documents. You can then specify a default version for each document. The default version of a document can be updated to a newer version or reverted to an older version of the document. When you change the content of a document, Systems Manager automatically increments the version of the document. You can retrieve or use any version of a document by specifying the document version in the console, CLI commands, or API calls.

**Customizing a document**
If you want to customize the steps and actions in a document, you can create your own. The first time you use a document to perform an action on an instance, the system stores the document with your AWS account. For more information about how to create an SSM document, see Creating Systems Manager documents (p. 1084).

**Tagging a document**

You can tag your documents to help you quickly identify one or more documents based on the tags you've assigned to them. For example, you can tag documents for specific environments, departments, users, groups, or periods. You can also restrict access to documents by creating an IAM policy that specifies the tags that a user or group can access. For more information, see Tagging Systems Manager documents (p. 1164).

**Sharing a document**

You can make your documents public or share them with specific AWS accounts in the same AWS Region. Sharing documents between accounts can be useful if, for example, you want all of the EC2 instances that you supply to customers or employees to have the same configuration. In addition to keeping applications or patches on the instances up-to-date, you might want to restrict customer instances from certain activities. Or you might want to ensure that the instances used by employee accounts throughout your organization are granted access to specific internal resources. For more information, see Sharing SSM documents (p. 1091).

**SSM document quotas**

For information about SSM document quotas, see Systems Manager service quotas in the Amazon Web Services General Reference.

**Topics**

- SSM document schemas and features (p. 1033)
- SSM document syntax (p. 1046)
- SSM document plugin reference (p. 1052)
- Creating Systems Manager documents (p. 1084)
- Sharing SSM documents (p. 1091)
- Running SSM documents from remote locations (p. 1099)

**SSM document schemas and features**

SSM documents currently use the following schema versions.

- Documents of type **Command** can use schema version 1.2, 2.0, and 2.2. If you are currently using schema 1.2 documents, we recommend that you create documents that use schema version 2.2.
- Documents of type **Policy** must use schema version 2.0 or later.
- Documents of type **Automation** must use schema version 0.3.
- You can create documents in JSON or YAML.

By using the latest schema version for **Command** and **Policy** documents, you can take advantage of the following features.
Schema version 2.2 document features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document editing</td>
<td>Documents can now be updated. With version 1.2, any update to a document required that you save it with a different name.</td>
</tr>
<tr>
<td>Automatic versioning</td>
<td>Any update to a document creates a new version. This is not a schema version, but a version of the document.</td>
</tr>
<tr>
<td>Default version</td>
<td>If you have multiple versions of a document, you can specify which version is the default document.</td>
</tr>
<tr>
<td>Sequencing</td>
<td>Plugins or steps in a document run in the order that you specified.</td>
</tr>
<tr>
<td>Cross-platform support</td>
<td>Cross-platform support enables you to specify different operating systems for different plugins within the same SSM document. Cross-platform support uses the precondition parameter within a step.</td>
</tr>
</tbody>
</table>

**Note**
You must keep SSM Agent on your instances updated with the latest version to use new Systems Manager features and SSM document features. For more information, see Update SSM Agent by using Run Command (p. 835).

The following table lists the differences between major schema versions.

<table>
<thead>
<tr>
<th>Version 1.2</th>
<th>Version 2.2 (latest version)</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtimeConfig</td>
<td>mainSteps</td>
<td>In version 2.2, the mainSteps section replaces runtimeConfig. The mainSteps section enables Systems Manager to run steps in sequence.</td>
</tr>
<tr>
<td>properties</td>
<td>inputs</td>
<td>In version 2.2, the inputs section replaces the properties section. The inputs section accepts parameters for steps.</td>
</tr>
<tr>
<td>commands</td>
<td>runCommand</td>
<td>In version 2.2, the inputs section takes the runCommand parameter instead of the commands parameter.</td>
</tr>
<tr>
<td>id</td>
<td>action</td>
<td>In version 2.2, Action replaces ID. This is just a name change.</td>
</tr>
<tr>
<td>not applicable</td>
<td>name</td>
<td>In version 2.2, name is any user-defined name for a step.</td>
</tr>
</tbody>
</table>
Using the precondition parameter

With schema version 2.2 or later, you can use the `precondition` parameter to specify the target operating system for each plugin. The `precondition` parameter supports `platformType` and a value of either Windows or Linux.

For documents that use schema version 2.2 or later, if `precondition` is not specified, each plugin is either run or skipped based on the plugin's compatibility with the operating system. For documents that use schema 2.0 or earlier, incompatible plugins throw an error.

For example, in a schema version 2.2 document, if `precondition` is not specified and the `aws:runShellScript` plugin is listed, then the step runs on Linux instances, but the system skips it on Windows Server instances because the `aws:runShellScript` is not compatible with Windows Server instances. However, for a schema version 2.0 document, if you specify the `aws:runShellScript` plugin, and then run the document on a Windows Server instances, the execution fails. You can see an example of the precondition parameter in an SSM document later in this section.

Schema version 2.2

Top-level elements

The following example shows the top-level elements of an SSM document using schema version 2.2.

YAML

```yaml
---
schemaVersion: "2.2"
description: A description of the document.
parameters:
  parameter 1:
    property 1: "value"
    property 2: "value"
  parameter 2:
    property 1: "value"
    property 2: "value"
mainSteps:
- action: Plugin name
  name: A name for the step.
  inputs:
    input 1: "value"
    input 2: "value"
    input 3: "{{ parameter 1 }}"
```

JSON

```json
{
    "schemaVersion": "2.2",
    "description": "A description of the document.",
    "parameters": {
        "parameter 1": {
            "property 1": "value",
            "property 2": "value"
        },
        "parameter 2": {
            "property 1": "value",
            "property 2": "value"
        }
    },
    "mainSteps": [
        {
```
"action": "Plugin name",
"name": "A name for the step.",
"inputs": {
  "input 1": "value",
  "input 2": "value",
  "input 3": "{{ parameter 1 }}"
}
}
]

---

schemaVersion: "2.2"
description: "Example document"
parameters:
  Message:
    type: "String"
description: "Example parameter"
default: "Hello World"
mainSteps:
  - action: "aws:runPowerShellScript"
    name: "example"
    inputs:
      runCommand:
        - "Write-Output {{Message}}"

---

{ "schemaVersion": "2.2",
  "description": "Example document",
  "parameters": {
    "Message": {
      "type": "String",
      "description": "Example parameter",
      "default": "Hello World"
    }
  },
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "example",
      "inputs": {
        "runCommand": [
          "Write-Output {{Message}}"
        ]
      }
    }
  ]
}

---

Schema version 2.2 example

The following example uses the *aws:runPowerShellScript* plugin to run a PowerShell command on the target instances.

YAML

```yaml
---

schemaVersion: "2.2"
description: "Example document"
parameters:
  Message:
    type: "String"
description: "Example parameter"
default: "Hello World"
mainSteps:
  - action: "aws:runPowerShellScript"
    name: "example"
    inputs:
      runCommand:
        - "Write-Output {{Message}}"

---

JSON

```json

{ "schemaVersion": "2.2",
  "description": "Example document",
  "parameters": {
    "Message": {
      "type": "String",
      "description": "Example parameter",
      "default": "Hello World"
    }
  },
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "example",
      " inputs": {
        "runCommand": [
          "Write-Output {{Message}}"
        ]
      }
    }
  ]
}

---

Schema version 2.2 precondition parameter example

1036
Schema version 2.2 provides cross-platform support. This means that within a single SSM document you can specify different operating systems for different plugins. Cross-platform support uses the `precondition` parameter within a step, as shown in the following example.

**YAML**

```yaml
---
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
```

**JSON**

```json
{
  "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"
        ]
      }
    }
  ]
}
```
Schema version 2.2 State Manager example

You can use the following SSM document with State Manager to download and install the ClamAV antivirus software. State Manager enforces a specific configuration, which means that each time the State Manager association is run, the system checks to see if the ClamAV software is installed. If not, State Manager reruns this document.

YAML

```yaml
---
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
```

JSON

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

Schema version 2.2 Inventory example

You can use the following SSM document with State Manager to collect inventory metadata about your instances.

YAML

```yaml
---
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 }}"

```json
{
  "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 }}"
    }
  }
]
**Schema version 2.2 AWS-ConfigureAWSPackage example**

The following example shows the AWS-ConfigureAWSPackage document. The mainSteps section includes the `aws:configurePackage` plugin in the `action` step.

**Note**

On Linux operating systems, only the AmazonCloudWatchAgent and AWSSupport-EC2Rescue packages are supported.

**YAML**

```yaml
---
schemaVersion: '2.2'
description: 'Install or uninstall the latest version or specified version of an AWS package. Available packages include the following: AWSPVDriver, AwsEnaNetworkDriver, 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-zA-Z)[a-zA-Z0-9-\_]\{0,39}\{[a-zA-Z0-9-\_]\{0,9}\}*[^a-zA-Z0-9-\_])$|^[a-zA-Z][a-zA-Z0-9-\_]{0,39}$"
  version:
    type: String
    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."
    default: latest
mainSteps:
  - action: aws:configurePackage
    name: configurePackage
    inputs:
      name: "{{ name }}"
      action: "{{ action }}"
      version: "{{ version }}"
```

**JSON**

```json
{
  "schemaVersion": "2.2",
  "description": "Install or uninstall the latest version or specified version of an AWS package. Available packages include the following: AWSPVDriver, AwsEnaNetworkDriver, 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"
      ]
    }
  }
```

1041
Schema version 1.2

The following example shows the top-level elements of a schema version 1.2 document.

```json
{
  "schemaVersion": "1.2",
  "description": "A description of the SSM 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"
      }
    }
  }
}
```

Schema version 1.2 aws:runShellScript example
The following example shows the AWS-RunShellScript SSM document. The runtimeConfig section includes the aws:runShellScript plugin.

```
{
    "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":"
        },
        "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]\[0-9\]{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 }}"
            }
        }
    }
}
```

Schema version 0.3

Top-level elements

The following example shows the top-level elements of a schema version 0.3 Automation document in JSON format.

```
{
    "description": "document-description",
    "schemaVersion": "0.3",
    "assumeRole": "{{assumeRole}}",
    "parameters": {
        "parameter-1": {
            "type": "String",
            "description": "parameter-1-description",
            "default": ""
        },
        "parameter-2": {
            "type": "String",
            "description": "parameter-2-description",
            "default": ""
        }
    }
}
```
YAML Automation document example

The following sample shows the contents of an Automation document, in YAML format. This working example of version 0.3 of the document schema also demonstrates the use of Markdown to format document descriptions.

description: >-

##Title: LaunchInstanceAndCheckState

-----

**Purpose**: This Automation document first launches an EC2 instance using the AMI ID provided in the parameter `imageId`. The second step of this document continuously checks the instance status check value for the launched instance until the status `ok` is returned.

##Parameters:

-----

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>assumeRole</td>
<td>String</td>
<td>(Optional) The ARN of the role that allows Automation to perform the actions on your behalf.</td>
<td>-</td>
</tr>
<tr>
<td>imageId</td>
<td>String</td>
<td>(Optional) The AMI ID to use for launching the instance. The default value uses the latest Amazon Linux AMI ID available.</td>
<td><code>{ssm::aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2}</code></td>
</tr>
</tbody>
</table>

schemaVersion: '0.3'

assumeRole: 'arn:aws:iam::111122223333::role/AutomationServiceRole'

parameters:
imageId:
type: String
default: '{ssm:/aws/service/ami-amazon-linux-latest/amzn-ami-hvm-x86_64-gp2}"
description: >-
(Optional) The AMI ID to use for launching the instance. The default value
uses the latest released Amazon Linux AMI ID.
tagValue:
type: String
default: 'LaunchedBySsmAutomation'
description: >-
(Optional) The tag value to add to the instance. The default value is
LaunchedBySsmAutomation.
instanceType:
type: String
default: t2.micro
description: >-
(Optional) The instance type to use for the instance. The default value is
t2.micro.
mainSteps:
  - name: LaunchEc2Instance
    action: 'aws:executeScript'
    outputs:
      - Name: payload
        Selector: $.Payload
        Type: StringMap
    inputs:
      Runtime: python3.6
      Handler: launch_instance
      Script: |
      InputPayload:
        image_id: '{imageId}'
        tag_value: '{tagValue}'
        instance_type: '{instanceType}'
      Attachment: launch.py
description: >-
**About This Step**

This step first launches an EC2 instance using the ```aws:executeScript```
action and the provided python script.
  - name: WaitForInstanceStatusOk
    action: 'aws:executeScript'
    inputs:
      Runtime: python3.6
      Handler: poll_instance
      Script: |-
      def poll_instance(events, context):
        import boto3
        import time
        ec2 = boto3.client('ec2')
        instance_id = events['InstanceId']
        print('[INFO] Waiting for instance status check to report ok', instance_id)
        instance_status = "null"
        while True:
          res = ec2.describe_instance_status(InstanceIds=[instance_id])
          if len(res['InstanceStatuses']) == 0:
            print("Instance status information is not available yet")
            time.sleep(5)
            continue
instance_status = res['InstanceStatuses'][0]['InstanceStatus']['Status']
print('[INFO] Polling to get status of the instance', instance_status)

if instance_status == 'ok':
    break

time.sleep(10)

    return {'Status': instance_status, 'InstanceId': instance_id}

InputPayload: '{{ LaunchEc2Instance.payload }}'
description: >-
**About This Step**

The python script continuously polls the instance status check value for
the instance launched in Step 1 until the ```ok``` status is returned.

files:
    launch.py:
        checksums:
            sha256: 18871b1311b295c43d0f...[truncated]...772da97b67e99d84d342ef4aEXAMPLE

SSM document syntax

The syntax of your document is defined by the schema version used to create it. We recommended
that you use schema version 2.2 or later for command documents. Automation documents use schema
version 0.3. Additionally, Automation documents support the use of Markdown, a markup language,
which allows you to add wiki-style descriptions to documents and individual steps within the document.
For more information on using Markdown, see Using Markdown in AWS.

The top-level elements provide the structure of the SSM document. The information in this topic
pertains to Command and Automation SSM documents.

Top-level elements

schemaVersion
The schema version to use.
Type: Version
Required: Yes
description
Information you provide to describe the purpose of the document.
Type: String
Required: No
parameters
A structure that defines the parameters the document accepts. For parameters that you reference
often, we recommend that you store those parameters in Systems Manager Parameter Store and
then reference them. You can reference String and StringList Parameter Store parameters in
this section of a document. You can't reference SecureString Parameter Store parameters in this
section of a document. For more information, see AWS Systems Manager Parameter Store (p. 198).
Type: Structure
The parameters structure accepts the following fields and values:
• type: (Required) Allowed values include the following: String, StringList, Boolean, Integer, MapList, and StringMap. To view examples of each type, see SSM document parameter type examples (p. 1049) in the next section.

• description: (Optional) A description of the parameter.

• default: (Optional) The default value of the parameter or a reference to a parameter in Parameter Store.

• allowedValues: (Optional) An array of values allowed for the parameter. Defining allowed values for the parameter validates the user input. If a user inputs a value that is not allowed, the execution fails to start.

YAML

```yaml
DirectoryType:
  type: String
  description: "(Required) The directory type to launch."
  default: AwsMad
  allowedValues:
  - AdConnector
  - AwsMad
  - SimpleAd
```

JSON

```json
"DirectoryType": {
  "type": "String",
  "description": "(Required) The directory type to launch.",
  "default": "AwsMad",
  "allowedValues": [
    "AdConnector",
    "AwsMad",
    "SimpleAd"
  ]
}
```

• allowedPattern: (Optional) A regular expression that validates whether the user input matches the defined pattern for the parameter. If the user input does not match the allowed pattern, the execution fails to start.

YAML

```yaml
InstanceId:
  type: String
  description: "(Required) The instance ID to target."
  allowedPattern: "^i-[a-z0-9]{8,17}$"
  default: ''
```

JSON

```json
"InstanceId": {
  "type": "String",
  "description": "(Required) The instance ID to target.",
  "allowedPattern": "^i-[a-z0-9]{8,17}$",
  "default": ""
}
```

• displayType: (Optional) Used to display either a textfield or a textarea in the AWS Management Console. textfield is a single-line text box. textarea is a multi-line text area.

• minItems: (Optional) The minimum number of items allowed.

• maxItems: (Optional) The maximum number of items allowed.
• **minChars**: (Optional) The minimum number of parameter characters allowed.
  • **maxChars**: (Optional) The maximum number of parameter characters allowed.

  Required: No

**runtimeConfig**

(Schema version 1.2 only) The configuration for the instance as applied by one or more Systems Manager plugins. Plugins are not guaranteed to run in sequence.

Type: Dictionary<string,PluginConfiguration>

Required: No

**mainSteps**

(Schema version 0.3, 2.0, and 2.2 only) An object that can include multiple steps (plugins). Plugins are defined within steps. Steps run in sequential order as listed in the document.

Type: Dictionary<string,PluginConfiguration>

Required: Yes

**outputs**

(Schema version 0.3 only) Data generated by the execution of this document that can be used in other processes. For example, if your document creates a new AMI, you might specify "CreateImage.ImageId" as the output value, and then use this output to create new instances in a subsequent automation execution. For more information about outputs, see Working with inputs and outputs (p. 378).

Type: Dictionary<string,OutputConfiguration>

Required: No

**files**

(Schema version 0.3 only) The script files (and their checksums) attached to the document and run during an automation execution. Applies only to documents that include the `aws:executeScript` action and for which attachments have been specified in one or more steps.

For script runtime support, Automation documents currently support scripts for Python 3.6, Python 3.7, and PowerShell Core 6.0. For more information about including scripts in Automation documents, see Creating Automation documents that run scripts (p. 358) and Walkthrough: Using Document Builder to create a custom Automation document (p. 604).

When you create an Automation document, or playbook, you specify attachment files using the `--attachments` option (for AWS CLI) or `Attachments` (for API and SDK). You can specify the file location for both local files and files stored in Amazon S3 buckets.

**YAML**

```yaml
---
files:
  launch.py:
    checksums:
      sha256: 18871b1311b295c43d0f...[truncated]...772da97b67e99d84d342ef4aEXAMPLE
```

**JSON**

```json
"files": {
  "launch.py": {
    "checksums": {
```
SSM document syntax

Type: Dictionary<string,FilesConfiguration>
Required: No

SSM document parameter type examples

Parameter types in SSM documents are static. This means the parameter type can't be changed after it is defined. When using parameters with SSM document plugins, the type of a parameter can't be dynamically changed within a plugin's input. For example, you can't reference an Integer parameter within the runCommand input of the aws:runShellScript plugin because this input accepts a string or list of strings. To use a parameter for a plugin input, the parameter type must match the accepted type. For example, you must specify a Boolean type parameter for the allowDowngrade input of the aws:updateSsmAgent plugin. If your parameter type doesn't match the input type for a plugin, the SSM document fails to validate and the system doesn't create the document.

When using parameters with SSM Automation actions, parameter types aren't validated when you create the SSM document in most cases. Only when you use the aws:runCommand action are parameter types validated when you create the SSM document. In all other cases, the parameter validation occurs during the automation execution when an action's input is verified before running the action. For example, if your input parameter is a String and you reference it as the value for the MaxInstanceCount input of the aws:runInstances action, the SSM document is created. However, when running the document, the automation fails while validating the aws:runInstances action because the MaxInstanceCount input requires an Integer.

The following are examples of each parameter type.

String
A sequence of zero or more Unicode characters wrapped in quotation marks. For example, "i-1234567890abcdef0". Use backslashes to escape.

YAML

```yaml
---
InstanceId:
  type: String
  description: "(Required) The target EC2 instance ID."
```

JSON

```json
"InstanceId":{
  "type":"String",
  "description":"(Required) The target EC2 instance ID."
}
```

StringList
A list of String items separated by commas. For example, ["cd ~","pwd"].

YAML

```yaml
---
commands:
  type: StringList
```
description: "(Required) Specify a shell script or a command to run."
minItems: 1
displayType: textarea

JSON

"commands":{
  "type":"StringList",
  "description":"(Required) Specify a shell script or a command to run.",
  "minItems":1,
  "displayType":"textarea"
}

Boolean

Accepts only true or false. Does not accept "true" or 0.

YAML

---
canRun:
  type: Boolean
  description: '
  default: true

JSON

"canRun": {
  "type": "Boolean",
  "description": "",
  "default": true
}

Integer

Integral numbers. Doesn't accept decimal numbers, for example 3.14159, or numbers wrapped in quotation marks, for example "3".

YAML

---
timeout:
  type: Integer
  description: The type of action to perform.
  default: 100

JSON

"timeout": {
  "type": "Integer",
  "description": "The type of action to perform.",
  "default": 100
}

StringMap

A mapping of keys to values. A key can only be a string. For example, {"Env": "Prod"}.

YAML

---

1050
notificationConfig:
  type: StringMap
  description: The configuration for events to be notified about
  default:
    NotificationType: Command
    NotificationEvents:
      - Failed
    NotificationArn: "$dependency.topicArn"
  maxChars: 150

JSON

"notificationConfig" : {
  "type" : "StringMap",
  "description" : "The configuration for events to be notified about",
  "default" : {
    "NotificationType" : "Command",
    "NotificationEvents" : ["Failed"],
    "NotificationArn" : "$dependency.topicArn"
  },
  "maxChars" : 150
}

MapList

A list of StringMap items.

YAML

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

JSON

"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
}
SSM document plugin reference

This reference describes the plugins that you can specify in an AWS Systems Manager (SSM) document. These plugins cannot be used in SSM Automation documents which use Automation actions. For information about AWS Systems Manager Automation actions, see Systems Manager Automation actions reference (p. 386).

Systems Manager determines the actions to perform on a managed instance by reading the contents of an SSM document. Each document includes a code-execution section. Depending on the schema version of your document, this code-execution section can include one or more plugins or steps. For the purpose of this Help topic, plugins and steps are called plugins. This section includes information about each of the Systems Manager plugins. For more information about documents, including information about creating documents and the differences between schema versions, see AWS Systems Manager documents (p. 1031).

Note
Some of the plugins described here run only on either Windows Server instances or Linux instances. Platform dependencies are noted for each plugin.

Contents

- aws:applications (p. 1052)
- aws:cloudWatch (p. 1054)
- aws:configureDocker (p. 1060)
- aws:configurePackage (p. 1061)
- aws:domainJoin (p. 1063)
- aws:downloadContent (p. 1065)
- aws:psModule (p. 1068)
- aws:refreshAssociation (p. 1070)
- aws:runDockerAction (p. 1071)
- aws:runDocument (p. 1073)
- aws:runPowerShellScript (p. 1074)
- aws:runShellScript (p. 1076)
- aws:softwareInventory (p. 1078)
- aws:updateAgent (p. 1080)
- aws:updateSsmAgent (p. 1082)

aws:applications

Install, repair, or uninstall applications on an EC2 instance. This plugin only runs on Windows Server operating systems. For more information, see AWS Systems Manager documents (p. 1031).

Syntax

Schema 2.2

YAML

```yaml
---
schemaVersion: '2.2'
description: aws:applications plugin
parameters:
  source:
    description: "(Required) Source of msi."
type: String
```
mainSteps:
- action: aws:applications
  name: example
  inputs:
    action: Install
    source: "{{ source }}"

JSON

```json
{
  "schemaVersion":"2.2",
  "description":"aws:applications",
  "parameters":{
    "source":{
      "description": "(Required) Source of msi.",
      "type": "String"
    }
  },
  "mainSteps": [
    {
      "action":"aws:applications",
      "name":"example",
      "inputs":{
        "action":"Install",
        "source":"{{ source }}"
      }
    }
  ]
}
```

Schema 1.2

YAML

```yaml
---
runtimeConfig:
  aws:applications:
    properties:
      - id: 0.aws:applications
        action: "{{ action }}"
        parameters: "{{ parameters }}"
        source: "{{ source }}"
        sourceHash: "{{ sourceHash }}"
```

JSON

```json
{
  "runtimeConfig":{
    "aws:applications":{
      "properties":[
        {"id":"0.aws:applications",
        "action":"{{ action }}",
        "parameters":"{{ parameters }}",
        "source":"{{ source }}",
        "sourceHash":"{{ sourceHash }}"}
      ]
    }
  }
}
```
Properties

action
The action to take.
Type: Enum
Valid values: Install | Repair | Uninstall
Required: Yes

parameters
The parameters for the installer.
Type: String
Required: No

source
The URL of the .msi file for the application.
Type: String
Required: Yes

sourceHash
The SHA256 hash of the .msi file.
Type: String
Required: No

aws:cloudWatch
Export data from Windows Server to Amazon CloudWatch or Amazon CloudWatch Logs and monitor the data using CloudWatch metrics. This plugin only runs on Windows Server operating systems. For more information about configuring CloudWatch integration with Amazon EC2, see Sending Logs, Events, and Performance Counters to Amazon CloudWatch. For more information about documents, see AWS Systems Manager documents (p. 1031).

Important
This plugin has been deprecated. The unified CloudWatch agent has replaced SSM Agent as the tool for sending log data to Amazon CloudWatch Logs. We recommend using only the unified CloudWatch agent for your log collection processes. For more information, see the following topics:

- Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135)
- Migrate Windows Server instance log collection to the CloudWatch agent (p. 1136)
- Collect metrics from Amazon Elastic Compute Cloud instances and on-premises servers with the CloudWatch agent in the Amazon CloudWatch User Guide

You can export and monitor the following data types:

ApplicationEventLog
Sends application event log data to CloudWatch Logs.
CustomLogs

Sends any text-based log file to CloudWatch Logs. The CloudWatch plugin creates a fingerprint for log files. The system then associates a data offset with each fingerprint. The plugin uploads files when there are changes, records the offset, and associates the offset with a fingerprint. This method is used to avoid a situation where a user enables the plugin, associates the service with a directory that contains a large number of files, and the system uploads all of the files.

**Warning**

Be aware that if your application truncates or attempts to clean logs during polling, any logs specified for LogDirectoryPath can lose entries. If, for example, you want to limit log file size, create a new log file when that limit is reached, and then continue writing data to the new file.

ETW

Sends Event Tracing for Windows (ETW) data to CloudWatch Logs.

IIS

Sends IIS log data to CloudWatch Logs.

PerformanceCounter

Sends Windows performance counters to CloudWatch. You can select different categories to upload to CloudWatch as metrics. For each performance counter that you want to upload, create a **PerformanceCounter** section with a unique ID (for example, "PerformanceCounter2", "PerformanceCounter3", and so on) and configure its properties.

**Note**

If the SSM Agent or the CloudWatch plugin is stopped, performance counter data is not logged in CloudWatch. This behavior is different than custom logs or Windows Event logs. Custom logs and Windows Event logs preserve performance counter data and upload it to CloudWatch after SSM Agent or the CloudWatch plugin is available.

SecurityEventLog

Sends security event log data to CloudWatch Logs.

SystemEventLog

Sends system event log data to CloudWatch Logs.

You can define the following destinations for the data:

**CloudWatch**

The destination where your performance counter metric data is sent. You can add more sections with unique IDs (for example, "CloudWatch2", "CloudWatch3", and so on), and specify a different Region for each new ID to send the same data to different locations.

**CloudWatchLogs**

The destination where your log data is sent. You can add more sections with unique IDs (for example, "CloudWatchLogs2", "CloudWatchLogs3", and so on), and specify a different Region for each new ID to send the same data to different locations.

**Syntax**

```
"runtimeConfig": {
  "aws:cloudWatch": {
    "settings": {
      "startType": "{{ status }}"
    },
```

1055
"properties": "{{ properties }}"
}

**Settings and properties**

**AccessKey**

Your access key ID. This property is required unless you launched your instance using an IAM role. This property cannot be used with SSM.

Type: String

Required: No

**CategoryName**

The performance counter category from Performance Monitor.

Type: String

Required: Yes

**CounterName**

The name of the performance counter from Performance Monitor.

Type: String

Required: Yes

**CultureName**

The locale where the timestamp is logged. If **CultureName** is blank, it defaults to the same locale currently used by your Windows Server instance.

Type: String

Valid values: For a list of supported values, see National Language Support (NLS) on the Microsoft website. Note that the div, div-MV, hu, and hu-HU values are not supported.

Required: No

**DimensionName**

A dimension for your Amazon CloudWatch metric. If you specify **DimensionName**, you must specify **DimensionValue**. These parameters provide another view when listing metrics. You can use the same dimension for multiple metrics so that you can view all metrics belonging to a specific dimension.

Type: String

Required: No

**DimensionValue**

A dimension value for your Amazon CloudWatch metric.

Type: String

Required: No

**Encoding**

The file encoding to use (for example, UTF-8). Use the encoding name, not the display name.

Type: String
Valid values: For a list of supported values, see Encoding Class in the MSDN Library.

Required: Yes

Filter

The prefix of log names. Leave this parameter blank to monitor all files.

Type: String

Valid values: For a list of supported values, see the FileSystemWatcherFilter Property in the MSDN Library.

Required: No

Flows

Each data type to upload, along with the destination for the data (CloudWatch or CloudWatch Logs). For example, to send a performance counter defined under "Id": "PerformanceCounter", enter "PerformanceCounter,CloudWatch". Similarly, to send the custom log, ETW log, and system log to the CloudWatch Logs destination defined under "Id": "ETW", enter "ETW,CloudWatchLogs". In addition, you can send the same performance counter or log file to more than one destination. For example, to send the application log to two different destinations that you defined under "Id": "CloudWatchLogs" and "Id": "CloudWatchLogs2", enter "ApplicationEventLog,(CloudWatchLogs, CloudWatchLogs2)".

Type: String

Valid values (source): ApplicationEventLog | CustomLogs | ETW | PerformanceCounter | SystemEventLog | SecurityEventLog

Valid values (destination): CloudWatch | CloudWatchLogs | CloudWatchLogs | CloudWatch

Required: Yes

FullName

The full name of the component.

Type: String

Required: Yes

Id

Identifies the data source or destination. This identifier must be unique within the configuration file.

Type: String

Required: Yes

InstanceName

The name of the performance counter instance. Do not use an asterisk (*) to indicate all instances because each performance counter component only supports one metric. You can, however use _Total.

Type: String

Required: Yes

Levels

The types of messages to send to Amazon CloudWatch.

Type: String
Valid values:
- **1** - Only error messages uploaded.
- **2** - Only warning messages uploaded.
- **4** - Only information messages uploaded.

Note that you can add values together to include more than one type of message. For example, **3** means that error messages (**1**) and warning messages (**2**) are included. A value of **7** means that error messages (**1**), warning messages (**2**), and informational messages (**4**) are included.

**Required:** Yes

**Note**
Windows Security Logs should set Levels to 7.

**LineCount**
The number of lines in the header to identify the log file. For example, IIS log files have virtually identical headers. You could enter **3**, which would read the first three lines of the log file's header to identify it. In IIS log files, the third line is the date and time stamp, which is different between log files.

**Type:** Integer

**Required:** No

**LogDirectoryPath**
For CustomLogs, the path where logs are stored on your EC2 instance. For IIS logs, the folder where IIS logs are stored for an individual site (for example, C:\inetpub\logs\LogFiles\W3SVCn). For IIS logs, only W3C log format is supported. IIS, NCSA, and Custom formats are not supported.

**Type:** String

**Required:** Yes

**LogGroup**
The name for your log group. This name is displayed on the Log Groups screen in the CloudWatch console.

**Type:** String

**Required:** Yes

**LogName**
The name of the log file.
1. To find the name of the log, in Event Viewer, in the navigation pane, click **Applications and Services Logs**.
2. In the list of logs, right-click the log you want to upload (for example, Microsoft>Windows>Backup>Operational), and then click **Create Custom View**.
3. In the Create Custom View dialog box, click the **XML** tab. The LogName is in the <Select Path=> tag (for example, Microsoft-Windows-Backup). Copy this text into the LogName parameter.

**Type:** String

**Valid values:** Application | Security | System | Microsoft-Windows-WinINet/Analytic

**Required:** Yes

**LogStream**
The destination log stream. If you use `{instance_id}`, the default, the instance ID of this instance is used as the log stream name.
Type: String

Valid values: \{instance_id\}|\{hostname\}|\{ip_address\} <log_stream_name>

If you enter a log stream name that doesn't already exist, CloudWatch Logs automatically creates it for you. You can use a literal string or predefined variables \{instance_id\}, \{hostname\}, \{ip_address\}, or a combination of all three to define a log stream name.

The log stream name specified in this parameter appears on the Log Groups > Streams for <YourLogStream> screen in the CloudWatch console.

Required: Yes

**MetricName**

The CloudWatch metric that you want performance data to appear under.

**Note**

Don't use special characters in the name. If you do, the metric and associated alarms might not work.

Type: String

Required: Yes

**NameSpace**

The metric namespace where you want performance counter data to be written.

Type: String

Required: Yes

**PollInterval**

How many seconds must elapse before new performance counter and log data is uploaded.

Type: Integer

Valid values: Set this to 5 or more seconds. Fifteen seconds (00:00:15) is recommended.

Required: Yes

**Region**

The AWS Region where you want to send log data. Although you can send performance counters to a different Region from where you send your log data, we recommend that you set this parameter to the same Region where your instance is running.

Type: String

Valid values: Regions IDs of the AWS Regions supported by both Systems Manager and CloudWatch Logs, such as us-east-2, eu-west-1, and ap-southeast-1. For lists of AWS Regions supported by each service, see Amazon CloudWatch Logs Service Endpoints and Systems Manager service endpoints in the Amazon Web Services General Reference.

Required: Yes

**SecretKey**

Your secret access key. This property is required unless you launched your instance using an IAM role.

Type: String

Required: No
**startType**

Enable or disable CloudWatch on the instance.

Type: String

Valid values: Enabled | Disabled

Required: Yes

**TimestampFormat**

The timestamp format you want to use. For a list of supported values, see [Custom Date and Time Format Strings](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/custom-date-and-time-format-strings) in the MSDN Library.

Type: String

Required: Yes

**TimeZoneKind**

Provides time zone information when no time zone information is included in your log's timestamp. If this parameter is left blank and if your timestamp doesn't include time zone information, CloudWatch Logs defaults to the local time zone. This parameter is ignored if your timestamp already contains time zone information.

Type: String

Valid values: Local | UTC

Required: No

**Unit**

The appropriate unit of measure for the metric.

Type: String


Required: Yes

**aws:configureDocker**

(Schema version 2.0 or later) Configure an instance to work with containers and Docker. This plugin is supported on Linux and Windows Server operating systems. For more information, see [AWS Systems Manager documents](https://docs.aws.amazon.com/systems-manager/latest/userguide/)

**Syntax**

**Schema 2.2**

**YAML**

```yaml
---
schemaVersion: '2.2'
description: aws:configureDocker
parameters:
  action:
```
description: "(Required) The type of action to perform."

type: String
default: Install
allowedValues:
- Install
- Uninstall

mainSteps:
- action: aws:configureDocker
  name: configureDocker
  inputs:
    action: "{{ action }}"

Inputs

action

The type of action to perform.

Type: Enum

Valid values: Install | Uninstall

Required: Yes

aws:configurePackage

(Schema version 2.0 or later) Install or uninstall a Distributor package. You can install the latest version, default version, or a version of the package you specify. Packages provided by AWS are also supported. This plugin runs on Windows Server and Linux operating systems, but not all the available packages are supported on Linux operating systems.

Available AWS packages for Windows Server include the following: AWSPVDriver, AWSNVMe, AwsEnaNetworkDriver, AwsVssComponents, AmazonCloudWatchAgent, and AWSSupport-EC2Rescue.
Available AWS packages for Linux operating systems include the following: AmazonCloudWatchAgent and AWSSupport-EC2Rescue.

For more information, see AWS Systems Manager documents (p. 1031).

Syntax

Schema 2.2

YAML

```yaml
---
schemaVersion: '2.2'
description: aws:configurePackage
parameters:
  name:
    description: "(Required) The name of the AWS package to install or uninstall."
    type: String
  action:
    description: "(Required) The type of action to perform."
    type: String
    default: Install
    allowedValues:
    - Install
    - Uninstall
mainSteps:
- action: aws:configurePackage
  name: configurePackage
  inputs:
    name: "{{ name }}"
    action: "{{ action }}"
```

JSON

```json
{
  "schemaVersion": "2.2",
  "description": "aws:configurePackage",
  "parameters": {
    "name": {
      "description": "(Required) The name of the AWS package to install or uninstall."
    },
    "action": {
      "description": "(Required) The type of action to perform."
    },
    "mainSteps": [
      {
        "name": "configurePackage",
        "inputs": {
          "name": "{{ name }}",
          "action": "{{ action }}"
        }
      }
    ]
}
```
Inputs

name
The name of the AWS package to install or uninstall. Available packages include the following: AWSPVDriver, AwsEnaNetworkDriver, AwsVssComponents, and AmazonCloudWatchAgent.

Type: String
Required: Yes

action
Install or uninstall a package.

Type: Enum
Valid values: Install | Uninstall
Required: Yes

installationType
The type of installation to perform. If you specify Uninstall and reinstall, the package is completely uninstalled, and then reinstalled. The application is unavailable until the reinstallation completes. If you specify In-place update, only new or changed files are added to the existing installation according you instructions you provide in an update script. The application remains available throughout the update process. Uninstall and reinstall is the default value.

Type:Enum
Valid values: Uninstall and reinstall | In-place update
Required: No

version
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
Required: No

aws:domainJoin
Join an EC2 instance to a domain. This plugin only runs on Windows Server operating systems. For more information, see AWS Systems Manager documents (p. 1031).

Syntax

Schema 2.2

YAML

---
schemaVersion: '2.2'
description: aws:domainJoin
parameters:
directoryId:
  description: "(Required) The ID of the directory."
type: String
directoryName:
  description: "(Required) The name of the domain."
  type: String
mainSteps:
- action: aws:domainJoin
  name: domainJoin
  inputs:
    directoryId: "{{ directoryId }}"
    directoryName: "{{ directoryName }}"
**Properties**

**directoryId**
- The ID of the directory.
- Type: String
- Required: Yes
- Example: "directoryId": "d-1234567890"

**directoryName**
- The name of the domain.
- Type: String
- Required: Yes
- Example: "directoryName": "example.com"

**directoryOU**
- The organizational unit (OU).
- Type: String
- Required: No
- Example: "directoryOU": "OU=test,DC=example,DC=com"

**dnsIpAddresses**
- The IP addresses of the DNS servers.
- Type: Array
- Required: No
- Example: "dnsIpAddresses": ["198.51.100.1","198.51.100.2"]

**Examples**

For examples, see Joining a Windows Server Instance to an AWS Directory Service Domain in the Amazon EC2 User Guide for Windows Instances.

**aws:downloadContent**

(Schema version 2.0 or later) Download SSM documents and scripts from remote locations. This plugin is supported on Linux and Windows Server operating systems.

**Syntax**

Schema 2.2

YAML

```yaml
---
```
schemaVersion: '2.2'
description: aws:downloadContent
parameters:
  sourceType:
    description: "(Required) The download source."
    type: String
  sourceInfo:
    description: "(Required) The information required to retrieve the content from
    the required source."
    type: String
mainSteps:
- action: aws:downloadContent
  name: downloadContent
  inputs:
    sourceType: "{{ sourceType }}"
    sourceInfo: "{{ sourceInfo }}"

Inputs

sourceType
The download source. Systems Manager currently supports the following source types for
downloading scripts and SSM documents: GitHub, S3, and SSMDocument.

Type: String
Required: Yes

sourceInfo
The information required to retrieve the content from the required source.

Type: StringMap
Required: Yes
For sourceType GitHub, specify the following:
- owner: The repository owner.
- repository: The name of the repository.
- path: The path to the file or directory you want to download.
- getOptions: Extra options to retrieve content from a branch other than master or from a specific commit in the repository. getOptions can be omitted if you are using the latest commit in the master branch.

This parameter uses the following format:
- branch:branch_name
  The default is master.
  "branch" is required only if your SSM document is stored in a branch other than master.
- commitID:commitID
  The default is head.
  To use the version of your SSM document in a commit other than the latest, specify the full commit ID. For example:

  "getOptions": "commitID:bbc1ddb94...b76d3bEXAMPLE",

- tokenInfo: The Systems Manager parameter (a SecureString parameter) where you store your GitHub access token information, in the format {{ssm-secure:secure-string-token-name}}.

  Note
  This tokenInfo field is the only SSM document plugin field that supports a SecureString parameter. SecureString parameters are not supported for any other fields, nor for any other SSM document plugins.

Example syntax:
```
{
  "owner":"TestUser",
  "repository": "GitHubTest",
  "path": "scripts/python/test-script",
  "getOptions": "branch:master",
  "tokenInfo": "{{ssm-secure:secure-string-token}}"
}
```

For sourceType S3, specify the following:
- path: The URL to the file or directory you want to download from Amazon S3.

Example syntax:
```
{
  "path": "https://s3.amazonaws.com/aws-executecommand-test/powershell/helloPowershell.ps1"
}
```

For sourceType SSMDocument, specify one of the following:
- name: The name and version of the document in the following format: name:version. Version is optional.

Example syntax:
```
{
  "name": "Example-RunPowerShellScript:3"
}
```
name: The ARN for the document in the following format:

{  "name":"arn:aws:ssm:us-east-2:3344556677:document/MySharedDoc"
}

destinationPath

An optional local path on the instance where you want to download the file. If you don't specify a path, the content is downloaded to a path relative to your command ID.

Type: String

Required: No

aws:psModule

Install PowerShell modules on an EC2 instance. This plugin only runs on Windows Server operating systems. For more information, see AWS Systems Manager documents (p. 1031).

Syntax

Schema 2.2

YAML

---

schemaVersion: '2.2'
description: aws:psModule
parameters:
  source:
    description: "(Required) The URL or local path on the instance to the application .zip file."
    type: String
mainSteps:
  - action: aws:psModule
    name: psModule
    inputs:
      source: "{{ source }}"

JSON

{
  "schemaVersion": "2.2",
  "description": "aws:psModule",
  "parameters": {
    "source": {
      "description": "(Required) The URL or local path on the instance to the application .zip file."
    },
  },
  "mainSteps": [
    {
      "action": "aws:psModule",
      "name": "psModule",
      "inputs": {
        "source": "{{ source }}"
      }
    }
  ]
}
Schema 1.2

YAML

```yaml
---
runtimeConfig:
  aws:psModule:
    properties:
      - runCommand: "{{ commands }}"
        source: "{{ source }}"
        sourceHash: "{{ sourceHash }}"
        workingDirectory: "{{ workingDirectory }}"
        timeoutSeconds: "{{ executionTimeout }}"
```

JSON

```json
{
  "runtimeConfig":{
    "aws:psModule":{
      "properties":[
        {
          "runCommand": "{{ commands }}",
          "source": "{{ source }}",
          "sourceHash": "{{ sourceHash }}",
          "workingDirectory": "{{ workingDirectory }}",
          "timeoutSeconds": "{{ executionTimeout }}"
        }
      ]
    }
  }
}
```

Properties

**runCommand**

The PowerShell command to run after the module is installed.

Type: StringList

Required: No

**source**

The URL or local path on the instance to the application .zip file.

Type: String

Required: Yes

**sourceHash**

The SHA256 hash of the .zip file.

Type: String
**timeoutSeconds**

The time in seconds for a command to be completed before it is considered to have failed.

*Type:* String

*Required:* No

**workingDirectory**

The path to the working directory on your instance.

*Type:* String

*Required:* No

---

### aws:refreshAssociation

(Schema version 2.0 or later) Refresh (force apply) an association on demand. This action will change the system state based on what is defined in the selected association or all associations bound to the targets. This plugin runs on Linux and Microsoft Windows Server operating systems. For more information, see AWS Systems Manager documents (p. 1031).

**Syntax**

**Schema 2.2**

**YAML**

```yaml
---
schemaVersion: '2.2'
description: aws:refreshAssociation
parameters:
  associationIds:
    description: "(Optional) List of association IDs. If empty, all associations bound to the specified target are applied."
    type: StringList
mainSteps:
- action: aws:refreshAssociation
  name: refreshAssociation
  inputs:
    associationIds:
    - "{{ associationIds }}"
```

**JSON**

```json
{
  "schemaVersion": "2.2",
  "description": "aws:refreshAssociation",
  "parameters": {
    "associationIds": {
      "description": "(Optional) List of association IDs. If empty, all associations bound to the specified target are applied."
    },
    "type": "StringList"
  },
  "mainSteps": [
    {
      "action": "aws:refreshAssociation",
      "name": "refreshAssociation",
```
"inputs": {
  "associationIds": [ 
    "{{ associationIds }}"
  ]
}
}

Inputs

associationIds

List of association IDs. If empty, all associations bound to the specified target are applied.

Type: StringList

Required: No

aws:runDockerAction

(Schema version 2.0 or later) Run Docker actions on containers. This plugin runs on Linux and Microsoft Windows Server operating systems. For more information, see AWS Systems Manager documents (p. 1031).

Syntax

Schema 2.2

YAML

```yaml
---
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 }}"
```

JSON

```json
{
  "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 \}"

}]

Inputs

action
The type of action to perform.
Type: String
Required: Yes

cpuShares
The container CPU shares (relative weight).
Type: String
Required: No

volume
The container volume mounts.
Type: StringList
aws:runDocument

(Schema version 2.0 or later) Runs SSM documents stored in Systems Manager or on a local share. You can use this plugin with the aws:downloadContent (p. 1065) plugin to download an SSM document from a remote location to a local share, and then run it. This plugin is supported on Linux and Windows Server operating systems.

Syntax

Schema 2.2

YAML

```yaml
---
schemaVersion: '2.2'
description: aws:runDocument
parameters:
documentType:
  description: "(Required) The document type to run."
type: String
allowedValues:
- LocalPath
- SSMDocuments
mainSteps:
- action: aws:runDocument
  name: runDocument
  inputs:
    documentType: "{{{ documentType }}}"
```

JSON

```json
{
  "schemaVersion": "2.2",
  "description": "aws:runDocument",
  "parameters": {
    "documentType": {
```
"description": "(Required) The document type to run.",
"type": "String",
"allowedValues": [
  "LocalPath",
  "SSMDocument"
]
},
"mainSteps": [
  {
    "action": "aws:runDocument",
    "name": "runDocument",
    "inputs": {
      "documentType": "{{ documentType }}"
    }
  }
]}

## Inputs

### documentType

The document type to run. You can run local documents (LocalPath) or documents stored in Systems Manager (SSMDocument).

Type: String

Required: Yes

### documentPath

The path to the document. If documentType is LocalPath, then specify the path to the document on the local share. If documentType is SSMDocument, then specify the name of the document.

Type: String

Required: No

### documentParameters

Parameters for the document.

Type: StringMap

Required: No

## aws:runPowerShellScript

Run PowerShell scripts or specify the path to a script to run. This plugin runs on Microsoft Windows Server and Linux operating systems. For more information, see AWS Systems Manager documents (p. 1031).

### Syntax

**Schema 2.2**

**YAML**

```yaml
---
```
schemaVersion: '2.2'
description: aws:runPowerShellScript
parameters:
  commands:
    type: String
    description: "(Required) The commands to run or the path to an existing script on the instance."
    default: Write-Host "Hello World"
mainSteps:
  - action: aws:runPowerShellScript
    name: runPowerShellScript
    inputs:
      timeoutSeconds: '60'
      runCommand: 
        - "{{ commands }}"

JSON

{
  "schemaVersion": "2.2",
  "description": "aws:runPowerShellScript",
  "parameters": {
    "commands": {
      "type": "String",
      "description": "(Required) The commands to run or the path to an existing script on the instance."
    },
    "default": "Write-Host "Hello World""
  },
  "mainSteps": [
    {
      "action": "aws:runPowerShellScript",
      "name": "runPowerShellScript",
      "inputs": {
        "timeoutSeconds": "60",
        "runCommand": [
          "{{ commands }}"
        ]
      }
    }
  ]
}

YAML

```yaml
---
runtimeConfig:
  aws:runPowerShellScript:
    properties:
      - id: 0.aws:runPowerShellScript
        runCommand: "{{ commands }}"
        workingDirectory: "{{ workingDirectory }}"
        timeoutSeconds: "{{ executionTimeout }}"

JSON

{
  "runtimeConfig": {
    "aws:runPowerShellScript": {
```
**Properties**

**runCommand**

Specify the commands to run or the path to an existing script on the instance.

*Type: StringList*

*Required: Yes*

**timeoutSeconds**

The time in seconds for a command to be completed before it is considered to have failed. When the timeout is reached, Systems Manager stops the command execution.

*Type: String*

*Required: No*

**workingDirectory**

The path to the working directory on your instance.

*Type: String*

*Required: No*

---

**aws:runShellScript**

Run Linux shell scripts or specify the path to a script to run. This plugin only runs on Linux operating systems. For more information, see AWS Systems Manager documents (p. 1031).

**Syntax**

**Schema 2.2**

**YAML**

```yaml
---
schemaVersion: '2.2'
description: aws:runShellScript
parameters:
  commands:
    type: String
    description: "(Required) The commands to run or the path to an existing script on the instance."
    default: echo Hello World
```
mainSteps:
- action: aws:runShellScript
  name: runShellScript
  inputs:
    timeoutSeconds: '60'
    runCommand:
      - "{{ commands }}"

JSON

```json
{
"schemaVersion": "2.2",
"description": "aws:runShellScript",
"parameters": {
  "commands": {
    "type": "String",
    "description": "(Required) The commands to run or the path to an existing script on the instance.",
    "default": "echo Hello World"
  }
},
"mainSteps": [
  {
    "action": "aws:runShellScript",
    "name": "runShellScript",
    "inputs": {
      "timeoutSeconds": "60",
      "runCommand": [
        "{{ commands }}"
      ]
    }
  }
]
}
```

Schema 1.2

YAML

```yaml
---
runtimeConfig:
  aws:runShellScript:
    properties:
      - runCommand: "{{ commands }}"
      workingDirectory: "{{ workingDirectory }}"
      timeoutSeconds: "{{ executionTimeout }}"

JSON

```json
{
"runtimeConfig":{
  "aws:runShellScript":{
    "properties":{
      "runCommand":"{{ commands }}",
      "workingDirectory":"{{ workingDirectory }}",
      "timeoutSeconds":"{{ executionTimeout }}"
    }
  }
}
```
Properties

**runCommand**

Specify the commands to run or the path to an existing script on the instance.

Type: StringList

Required: Yes

**timeoutSeconds**

The time in seconds for a command to be completed before it is considered to have failed. When the timeout is reached, Systems Manager stops the command execution.

Type: String

Required: No

**workingDirectory**

The path to the working directory on your instance.

Type: String

Required: No

**aws:softwareInventory**

(Schema version 2.0 or later) Gather metadata about applications, files, and configurations on your managed instances. This plugin runs on Linux and Microsoft Windows Server operating systems. For more information about collecting inventory, see AWS Systems Manager Inventory (p. 704).

Syntax

**Schema 2.2**

**YAML**

```yaml
---
mainSteps:
  - action: aws:softwareInventory
    name: collectSoftwareInventoryItems
    inputs:
      applications: "{{ applications }}"
      awsComponents: "{{ awsComponents }}"
      networkConfig: "{{ networkConfig }}"
      files: "{{ files }}"
      services: "{{ services }}"
      windowsRoles: "{{ windowsRoles }}"
      windowsRegistry: "{{ windowsRegistry }}"
      windowsUpdates: "{{ windowsUpdates }}"
      instanceDetailedInformation: "{{ instanceDetailedInformation }}"
      customInventory: "{{ customInventory }}"
```

**JSON**

```json
{
}
```
"mainSteps": [
  {
    "action": "aws:softwareInventory",
    "name": "collectSoftwareInventoryItems",
    "inputs": {
      "applications": "{{ applications }}",
      "awsComponents": "{{ awsComponents }}",
      "networkConfig": "{{ networkConfig }}",
      "files": "{{ files }}",
      "services": "{{ services }}",
      "windowsRoles": "{{ windowsRoles }}",
      "windowsRegistry": "{{ windowsRegistry }}",
      "windowsUpdates": "{{ windowsUpdates }}",
      "instanceDetailedInformation": "{{ instanceDetailedInformation }}",
      "customInventory": "{{ customInventory }}"
    }
  }
]

**Inputs**

**applications**

(Optional) Collect metadata for installed applications.

Type: String

Required: No

**awsComponents**

(Optional) Collect metadata for AWS components like amazon-ssm-agent.

Type: String

Required: No

**files**

(Optional, requires SSM Agent version 2.2.64.0 or later) Collect metadata for files, including file names, the time files were created, the time files were last modified and accessed, and file sizes, to name a few. For more information about collecting file inventory, see Working with file and Windows registry inventory (p. 711).

Type: String

Required: No

**networkConfig**

(Optional) Collect metadata for network configurations.

Type: String

Required: No

**windowsUpdates**

(Optional) Collect metadata for all Windows updates.

Type: String

Required: No
instanceDetailedInformation

(Optional) Collect more instance information than is provided by the default inventory plugin (aws:instanceInformation), including CPU model, speed, and the number of cores, to name a few.

Type: String
Required: No

services

(Optional, Windows OS only, requires SSM Agent version 2.2.64.0 or later) Collect metadata for service configurations.

Type: String
Required: No

windowsRegistry

(Optional, Windows OS only, requires SSM Agent version 2.2.64.0 or later) Collect Windows Registry keys and values. You can choose a key path and collect all keys and values recursively. You can also collect a specific registry key and its value for a specific path. Inventory collects the key path, name, type, and the value. For more information about collecting Windows Registry inventory, see Working with file and Windows registry inventory (p. 711).

Type: String
Required: No

windowsRoles

(Optional, Windows OS only, requires SSM Agent version 2.2.64.0 or later) Collect metadata for Microsoft Windows role configurations.

Type: String
Required: No

customInventory

(Optional) Collect custom inventory data. For more information about custom inventory, see Working with custom inventory (p. 736)

Type: String
Required: No

aws:updateAgent

Update the EC2Config service to the latest version or specify an older version. This plugin only runs on Microsoft Windows Server operating systems. For more information about the EC2Config service, see Configuring a Windows Instance Using the EC2Config Service. For more information about documents, see AWS Systems Manager documents (p. 1031).

Syntax

Schema 2.2

YAML

```yaml
---
schemaVersion: '2.2'
```
description: aws:updateAgent
mainSteps:
  - action: aws:updateAgent
    name: updateAgent
    inputs:
      agentName: Ec2Config
    source: https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json

JSON

```json
{
  "schemaVersion": "2.2",
  "description": "aws:updateAgent",
  "mainSteps": [
    {
      "action": "aws:updateAgent",
      "name": "updateAgent",
      "inputs": {
        "agentName": "Ec2Config",
        "source": "https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json"
      }
    }
  ]
}
```

Schema 1.2

YAML

```yaml
---
runtimeConfig:
  aws:updateAgent:
    properties:
      agentName: Ec2Config
      source: https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json
      allowDowngrade: "{{ allowDowngrade }}"
      targetVersion: "{{ version }}"
```

JSON

```json
{
  "runtimeConfig":{
    "aws:updateAgent":{
      "properties":{
        "agentName":"Ec2Config",
        "source":"https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json",
        "allowDowngrade":"{{ allowDowngrade }}",
        "targetVersion":"{{ version }}"
      }
    }
  }
}
```

Properties

agentName

EC2Config. This is the name of the agent that runs the EC2Config service.
Type: String
Required: Yes

**allowDowngrade**

Allow the EC2Config service to be downgraded to an earlier version. If set to false, the service can be upgraded to newer versions only (default). If set to true, specify the earlier version.

Type: Boolean
Required: No

**source**

The location where Systems Manager copies the version of EC2Config to install. You can't change this location.

Type: String
Required: Yes

**targetVersion**

A specific version of the EC2Config service to install. If not specified, the service will be updated to the latest version.

Type: String
Required: No

---

schemaVersion: '2.2'
description: aws:updateSsmAgent
parameters:
  allowDowngrade:
    default: 'false'
    description: "(Optional) Allow the Amazon SSM Agent service to be downgraded to an earlier version. If set to false, the service can be upgraded to newer versions only (default). If set to true, specify the earlier version."
    type: String
    allowedValues:
    - 'true'
    - 'false'
mainSteps:
- action: aws:updateSsmAgent
  name: updateSSMAgent
  inputs:
    agentName: amazon-ssm-agent
    source: https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-manifest.json
allowDowngrade: "{{ allowDowngrade }}"

### JSON

```json
{
    "schemaVersion": "2.2",
    "description": "aws:updateSsmAgent",
    "parameters": {
        "allowDowngrade": {
            "default": "false",
            "description": "(Required) Allow the Amazon SSM Agent service to be downgraded to an earlier version. If set to false, the service can be upgraded to newer versions only (default). If set to true, specify the earlier version."
        }
    },
    "mainSteps": [
        {
            "action": "aws:updateSsmAgent",
            "name": "updateSSMAgent",
            "inputs": {
                "agentName": "amazon-ssm-agent",
                "source": "https://s3.{Region}.amazonaws.com/amazon-ssm-{Region}/ssm-agent-manifest.json",
                "allowDowngrade": "{{ allowDowngrade }}"
            }
        }
    ]
}
```

### Schema 1.2

### YAML

```yaml
---
runtimeConfig:
  aws:updateSsmAgent:
    properties:
      - agentName: amazon-ssm-agent
        source: https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json
        allowDowngrade: "{{ allowDowngrade }}"
```

### JSON

```json
{
    "runtimeConfig":{
        "aws:updateSsmAgent":{
            "properties":{
                "agentName":"amazon-ssm-agent",
                "source":"https://s3.{Region}.amazonaws.com/aws-ssm-{Region}/manifest.json",
                "allowDowngrade":"{{ allowDowngrade }}"
            }
        }
    }
}
```
Creating Systems Manager documents

If the Systems Manager public documents don't perform all the actions you want to perform on your AWS resources, you can create your own SSM documents. When you create a new Command or Policy document, we recommend that you use schema version 2.2 or later so you can take advantage of the latest features, such as document editing, automatic versioning, sequencing, and more.

Writing SSM document content

To create your own SSM document content, it's important to understand the different schemas, features, plugins, and syntax available for SSM documents. We recommend becoming familiar with the following resources.

- Writing your own AWS Systems Manager documents
- SSM document syntax (p. 1046)
- SSM document schemas and features (p. 1033)
- SSM document plugin reference (p. 1052)
- Systems Manager Automation actions reference (p. 386)
AWS pre-defined SSM documents might perform some of the actions you require. You can call these documents by using the `aws:runDocument`, `aws:runCommand`, or `aws:executeAutomation` plugins within your custom SSM document, depending on the document type. You can also copy portions of those documents into a custom SSM document, and edit the content to meet your requirements.

**Tip**
When creating SSM document content, you might change the content and update your SSM document several times while testing. The following commands update the SSM document with your latest content, and update the document's default version to the latest version of the document.

**Note**
The Linux and Windows commands use the `jq` command line tool to filter the JSON response data.

**Linux**

```bash
```

**Windows**

```bash
latestDocVersion=$(aws ssm update-document --content file://C:path\to\file\documentContent.json --name "ExampleDocument" --document-format JSON --document-version "$LATEST" | jq -r '.DocumentDescription.LatestVersion')
```

**PowerShell**

```powershell
$content = Get-Content -Path "C:\path\to\file\documentContent.json" | Out-String
```

1085
Using SSM documents in State Manager Associations

If you create an SSM document for State Manager, you must associate the document with your managed instances after you add the document to the system. For more information, see Create an association (p. 870).

Keep in mind the following details when using SSM documents in State Manager associations.

- You can assign multiple documents to a target by creating different State Manager associations that use different documents.
- If you create a document with conflicting plugins (e.g., domain join and remove from domain), the last plugin run will be the final state. State Manager does not validate the logical sequence or rationality of the commands or plugins in your document.
- When processing documents, instance associations are applied first, and next tagged group associations are applied. If an instance is part of multiple tagged groups, then the documents that are part of the tagged group will not be run in any particular order. If an instance is directly targeted through multiple documents by its instance ID, there is no particular order of execution.
- If you change the default version of an SSM Policy document for State Manager, any association that uses the document will start using the new default version the next time Systems Manager applies the association to the instance.
- If you create an association using an SSM document that was shared with you, and then the owner stops sharing the document with you, your associations no longer have access to that document. However, if the owner shares the same SSM document with you again later, your associations automatically remap to it.

After writing your SSM document content, you can use your content to create an SSM document using one of the following methods.

Create SSM documents

- Create an SSM document (console) (p. 1086)
- Create an SSM document (command line) (p. 1087)
- Create an SSM document (API) (p. 1088)
- Creating composite documents (p. 1089)

Create an SSM document (console)

After you create the content for your custom SSM document, as described in Writing SSM document content (p. 1084), you can use the Systems Manager console to create an SSM document using your content.

To create an SSM document (console)

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Documents in the navigation pane.
3. Choose Create command or session.
4. Type a descriptive name for the document
5. (Optional) For Target type, specify the type of resources the document can run on.
6. In the Document type list, choose the type of document you want to create.
7. Delete the brackets in the **Content** field, and then paste the document content you created earlier.

8. (Optional) In the **Document tags** section, apply one or more tag key name/value pairs to the document.

   Tags are optional metadata that you assign to a resource. Tags enable you to categorize a resource in different ways, such as by purpose, owner, or environment. For example, you might want to tag a document to identify the type of tasks it runs, the type of operating systems it targets, and the environment it runs in. In this case, you could specify the following key name/value pairs:

   - **Key=TaskType,Value=MyConfigurationUpdate**
   - **Key=OS,Value=Ubuntu**
   - **Key=Environment,Value=Production**

9. Choose **Create document** to save the document.

## Create an SSM document (command line)

After you create the content for your custom SSM document, as described in Writing SSM document content (p. 1084), you can use the AWS CLI or AWS Tools for PowerShell to create an SSM document using your content. This is shown in the following command.

### Before you begin

Install and configure the AWS CLI or the AWS Tools for PowerShell, if you have not already. For information, see Install or upgrade AWS command line tools (p. 57).

#### Linux

```
aws ssm create-document \
  --content file://path/to/file/documentContent.json \
  --name "document-name" \
  --document-type "Command" \
  --tags "Key=tag-key,Value=tag-value"
```

#### Windows

```
aws ssm create-document ^
  --content file://C:\path\to\file\documentContent.json ^
  --name "document-name" ^
  --document-type "Command" ^
  --tags "Key=tag-key,Value=tag-value"
```

#### PowerShell

```
$json = Get-Content -Path "C:\path\to\file\documentContent.json" | Out-String
New-SSMDocument `
  -Content $json `
  -Name "document-name" `
  -DocumentType "Command" `
  -Tags "Key=tag-key,Value=tag-value"
```

document-name is the name of the SSM document you want to tag.

tag-key is the name of a custom key you supply. For example, Region or Quarter.

tag-value is the custom content for the value you want to supply for that key. For example, West or Q321.
If successful, the command has no output.

Create an SSM document (API)

After you create the content for your custom SSM document, as described in Writing SSM document content (p. 1084), you can use your preferred SDK to call the AWS Systems Manager CreateDocument API action to create an SSM document using your content. The JSON or YAML string for the Content request parameter is generally read from a file. The following sample functions create an SSM document using the SDKs for Python, Go, and Java.

Python

```python
import boto3

ssm = boto3.client('ssm')
filepath = '/path/to/file/documentContent.yaml'

def createDocumentApiExample():
    with open(filepath) as openFile:
        documentContent = openFile.read()
    createDocRequest = ssm.create_document(
        Content = documentContent,
        Name = 'createDocumentApiExample',
        DocumentType = 'Automation',
        DocumentFormat = 'YAML'
    )
    print(createDocRequest)
createDocumentApiExample()
```

Go

```go
package main

import (
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/ssm"
    "fmt"
    "io/ioutil"
    "log"
)

func main() {
    openFile, err := ioutil.ReadFile("/path/to/file/documentContent.yaml")
    if err != nil {
        log.Fatal(err)
    }
    documentContent := string(openFile)
    sesh := session.Must(session.NewSessionWithOptions(session.Options{
        SharedConfigState: session.SharedConfigEnable}))
    ssmClient := ssm.New(sesh)
    createDocRequest, err := ssmClient.CreateDocument(&ssm.CreateDocumentInput{
        Content: &documentContent,
        Name: aws.String("createDocumentApiExample"),
        DocumentType: aws.String("Automation"),
        DocumentFormat: aws.String("YAML"),
    })
    print(createDocRequest)
}
```
Creating composite documents

A composite SSM document is a custom document that performs a series of actions by running one or more secondary SSM documents. Composite documents promote *infrastructure as code* by enabling you to create a standard set of SSM documents for common tasks such as boot-strapping software or domain-joining instances. You can then share these documents across AWS accounts in the same AWS Region to reduce SSM document maintenance and ensure consistency.

For example, you can create a composite document that performs the following actions: 
1. Updates SSM Agent to the latest version.
2. Installs all whitelisted patches.
3. Installs antivirus software.
4. Downloads scripts from GitHub and runs them.

In this example, your custom SSM document includes the following plugins to perform these actions:

1. The `aws:runDocument` plugin to run the `AWS-UpdateSSMAgent` document, which updates SSM Agent to the latest version.
2. The `aws:runDocument` plugin to run the legacy `AWS-ApplyPatchBaseline` document, which installs all whitelisted patches.
3. The `aws:runDocument` plugin to run the `AWS-InstallApplication` document, which installs the antivirus software.
4. The `aws:downloadContent` plugin to download scripts from GitHub and run them.

Composite and secondary documents can be stored in Systems Manager, GitHub (public and private repositories), or Amazon S3. Composite documents and secondary documents can be created in JSON or YAML.

**Note**

Composite documents can only run to a maximum depth of three documents. This means that a composite document can call a child document; and that child document can call one last document.

**Create a composite document**

To create a composite document, add the `aws:runDocument` (p. 1073) plugin in a custom SSM document and specify the required inputs. The following is an example of a composite document that performs the following actions:

1. Runs the `aws:downloadContent` (p. 1065) plugin to download an SSM document from a GitHub public repository to a local directory called bootstrap. The SSM document is called `StateManagerBootstrap.yml` (a YAML document).
2. Runs the `aws:runDocument` plugin to run the `StateManagerBootstrap.yml` document. No parameters are specified.
3. Runs the `aws:runDocument` plugin to run the `AWS-ConfigureDocker` pre-defined SSM document. The specified parameters install Docker on the instance.

```json
{
  "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",
    ________________________________
Sharing SSM documents

You can share SSM documents privately or publicly with accounts in the same AWS Region. To privately share a document, you modify the document permissions and allow specific individuals to access it according to their Amazon Web Services (AWS) ID. To publicly share an SSM document, you modify the document permissions and specify All.

**Warning**

Use shared SSM documents only from trusted sources. When using any shared document, carefully review the contents of the document before using it so that you understand how it will change the configuration of your instance. For more information about shared document best practices, see Best practices for shared SSM documents (p. 1092).

**Limitations**

As you begin working with SSM documents, be aware of the following limitations.

- Only the owner can share a document.
- You must stop sharing a document before you can delete it. For more information, see Modify permissions for a shared SSM document (p. 1096).
- You can share a document with a maximum of 1000 AWS accounts. You can request an increase to this limit in the AWS Support Center. For **Limit type**, choose EC2 Systems Manager and describe your reason for the request.
- You can publicly share a maximum of five SSM documents. You can request an increase to this limit in the AWS Support Center. For **Limit type**, choose EC2 Systems Manager and describe your reason for the request.
- Documents can be shared with other accounts in the same AWS Region only. Cross-Region sharing is not supported.

Related topics

- For information about rebooting servers and instances when using Run Command to call scripts, see Rebooting managed instance from scripts (p. 843).
- For more information about creating an SSM document, see Creating Systems Manager documents (p. 1084).
- For more information about the plugins you can add to a custom SSM document, see SSM document plugin reference (p. 1052).
- If you simply want to run a document from a remote location (without creating a composite document), see Running SSM documents from remote locations (p. 1099).
Best practices for shared SSM documents

Review the following guidelines before you share or use a shared document.

Remove sensitive information

Review your SSM document carefully and remove any sensitive information. For example, verify that the document does not include your AWS credentials. If you share a document with specific individuals, those users can view the information in the document. If you share a document publicly, anyone can view the information in the document.

Restrict Run Command actions using an IAM user trust policy

Create a restrictive AWS Identity and Access Management (IAM) user policy for users who will have access to the document. The IAM policy determines which SSM documents a user can see in either the Amazon EC2 console or by calling ListDocuments using the AWS CLI or AWS Tools for Windows PowerShell. The policy also restricts the actions the user can perform with SSM documents. You can create a restrictive policy so that a user can only use specific documents. For more information, see Create non-Admin IAM users and groups for Systems Manager (p. 27) and Customer managed policy examples (p. 1117).

Use caution when using shared SSM documents

Review the contents of every document that is shared with you, especially public documents, to understand the commands that will be run on your instances. A document could intentionally or unintentionally have negative repercussions after it is run. If the document references an external network, review the external source before you use the document.

Send commands using the document hash

When you share a document, the system creates a Sha-256 hash and assigns it to the document. The system also saves a snapshot of the document content. When you send a command using a shared document, you can specify the hash in your command to ensure that the following conditions are true:

- You are running a command from the correct Systems Manager document
- The content of the document has not changed since it was shared with you.

If the hash does not match the specified document or if the content of the shared document has changed, the command returns an InvalidDocument exception. Note: The hash cannot verify document content from external locations.

Share an SSM document

You can share SSM documents by using the AWS Systems Manager console. You can also share SSM documents programmatically by calling the ModifyDocumentPermission API action using the AWS CLI, AWS Tools for Windows PowerShell, or the AWS SDK. Before you share a document, get the AWS account IDs of the people with whom you want to share. You will specify these account IDs when you share the document.
• Share a document (console) (p. 1093)
• Share a document (command line) (p. 1093)

Share a document (console)

Share a document

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon to open the navigation pane, and then choose Documents in the navigation pane.
3. In the documents list, choose the document you want to share, and then choose View details. On the Permissions tab, verify that you are the document owner. Only a document owner can share a document.
4. Choose Edit.
5. To share the command publicly, choose Public and then choose Save. To share the command privately, choose Private, enter the AWS account ID, choose Add permission, and then choose Save.

Share a document (command line)

The following procedure requires that you specify an AWS Region for your command line session.

1. Open the AWS CLI or AWS Tools for Windows PowerShell on your local computer and run the following command to specify your credentials.

   Linux
   ```bash
   aws config
   AWS Access Key ID: [your key]
   AWS Secret Access Key: [your key]
   Default region name: region
   Default output format [None]:
   ```

   Windows
   ```bash
   aws config
   AWS Access Key ID: [your key]
   AWS Secret Access Key: [your key]
   Default region name: region
   Default output format [None]:
   ```

   PowerShell
   ```powershell
   Set-AWSCredentials -AccessKey your key -SecretKey your key
   Set-DefaultAWSRegion -Region region
   ```

   region represents the identifier for an AWS Region supported by AWS Systems Manager, such as us-east-2 for the US East (Ohio) Region. For a list of supported region values, see the Region column in Systems Manager service endpoints in the Amazon Web Services General Reference.
2. Use the following command to list all of the SSM documents that are available for you. The list includes documents that you created and documents that were shared with you.

Linux

    aws ssm list-documents

Windows

    aws ssm list-documents

PowerShell

    Get-SSMDocumentList

3. Use the following command to get a specific document.

Linux

    aws ssm get-document \
      --name document name

Windows

    aws ssm get-document ^
      --name document name

PowerShell

    Get-SSMDocument `  
      -Name document name

4. Use the following command to get a description of the document.

Linux

    aws ssm describe-document \
      --name document name

Windows

    aws ssm describe-document ^
      --name document name

PowerShell

    Get-SSMDocumentDescription `  
      -Name document name

5. Use the following command to view the permissions for the document.

Linux

    aws ssm describe-document-permission \
      --name document name
--permission-type Share

Windows

```
aws ssm describe-document-permission ^
   --name document name ^
   --permission-type Share
```

PowerShell

```
Get-SSMDocumentPermission`
   -Name document name `
   -PermissionType Share
```

6. Use the following command to modify the permissions for the document and share it. You must be
the owner of the document to edit the permissions. This command privately shares the document
with a specific individual, based on that person's AWS account ID.

Linux

```
aws ssm modify-document-permission \\n   --name document name \\n   --permission-type Share \\n   --account-ids-to-add AWS account ID
```

Windows

```
aws ssm modify-document-permission ^
   --name document name ^
   --permission-type Share ^
   --account-ids-to-add AWS account ID
```

PowerShell

```
Edit-SSMDocumentPermission `
   -Name document name `
   -PermissionType Share `
   -AccountIdsToAdd AWS account ID
```

7. Use the following command to share a document publicly.

Linux

```
aws ssm modify-document-permission \\n   --name document name \\n   --permission-type Share \\n   --account-ids-to-add 'all'
```

Windows

```
aws ssm modify-document-permission ^
   --name document name ^
   --permission-type Share ^
   --account-ids-to-add "all"
```
Modify permissions for a shared SSM document

If you share a command, users can view and use that command until you either remove access to the SSM document or delete the SSM document. However, you cannot delete a document as long as it is shared. You must stop sharing it first and then delete it.

**Topics**
- Stop sharing a document (console) (p. 1096)
- Stop sharing a document (command line) (p. 1096)

**Stop sharing a document (console)**

2. In the navigation pane, choose **Documents**.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Documents** in the navigation pane.
3. In the documents list, choose the document you want to stop sharing, and then choose **View details**.
   On the **Permissions** tab, verify that you are the document owner. Only a document owner can stop sharing a document.
4. Choose **Edit**.
5. Choose **X** to delete the AWS account ID that should no longer have access to the command, and then choose **Save**.

**Stop sharing a document (command line)**

Open the AWS CLI or AWS Tools for Windows PowerShell on your local computer and run the following command to stop sharing a command.

**Linux**

```bash
aws ssm modify-document-permission \
    --name document name \
    --permission-type Share \
    --account-ids-to-remove 'AWS account ID'
```

**Windows**

```powershell
aws ssm modify-document-permission ^
    --name document name ^
    --permission-type Share ^
```
Using shared SSM documents

When you share an SSM document, the system generates an Amazon Resource Name (ARN) and assigns it to the command. If you select and run a shared document from the Amazon EC2 console, you do not see the ARN. However, if you want to run a shared SSM document from a command line application, you must specify a full ARN. You are shown the full ARN for an SSM document when you run the command to list documents.

**Note**
You are not required to specify ARNs for AWS public documents (documents that begin with AWS-*) or documents that you own.

Use a shared SSM document (command line)

To list all public SSM documents

Linux

```bash
aws ssm list-documents \  --filters Key=Owner,Values=Public
```

Windows

```bash
aws ssm list-documents \  --filters Key=Owner,Values=Public
```

PowerShell

```bash
$filter.Key = "Owner"
$filter.Values = "Public"
Get-SSMDocumentList \  -Filters @($filter)
```

To list private SSM documents that have been shared with you

Linux

```bash
aws ssm list-documents \  --filters Key=Owner,Values=Private
```

Windows

```bash
aws ssm list-documents \  --filters Key=Owner,Values=Private
```
PowerShell

```powershell
$filter.Key = "Owner"
$filter.Values = "Private"

Get-SSMDocumentList
 -Filters @{$filter}
```

To list all SSM documents available to you

Linux

```bash
aws ssm list-documents
```

Windows

```bash
aws ssm list-documents
```

PowerShell

```powershell
Get-SSMDocumentList
```

To get information about an SSM document that has been shared with you

Linux

```bash
aws ssm describe-document
```

Windows

```bash
aws ssm describe-document ^
```

PowerShell

```powershell
Get-SSMDocumentDescription
```

To run a shared SSM document

Linux

```bash
aws ssm send-command
 --instance-ids ID
```

Windows

```bash
aws ssm send-command ^
 --instance-ids ID
```
Running SSM documents from remote locations

You can run SSM documents from remote locations by using the AWS-RunDocument pre-defined SSM document. This document currently supports the following remote locations:

- GitHub repositories (public and private)
- Amazon S3
- Documents saved in Systems Manager

The following procedure describes how to run remote SSM documents by using the console. This procedure shows how to run the remote document by using Run Command, but you can also run remote documents by using State Manager or Automation.

Before you begin

Before you run a remote document, you must complete the following tasks.

- Create an SSM document and save it in a remote location. For more information, see Creating Systems Manager documents (p. 1084)
- If you plan to run a remote document that is stored in a private GitHub repository, then you must create a Systems Manager SecureString parameter for your GitHub security access token. You can't access a remote document in a private GitHub repository by manually passing your token over SSH. The access token must be passed as a Systems Manager SecureString parameter. For more information about creating a SecureString parameter, see Creating Systems Manager parameters (p. 249).

Run a remote document (console)

To run a remote document

2. In the navigation pane, choose Run Command.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Run Command.
3. Choose Run command.
5. In Command parameters, for Source Type, choose an option.
   - If you choose GitHub, specify Source Info information in the following format:

   ```json
   {
   "owner": "owner_name",
   "repository": "repository_name",
   "branch": "branch_name",
   "path": "path_to_document",
   }"
"tokenInfo": "{\{ssm-secure:SecureString_parameter_name\}}"
}

For example:

{
   "owner": "TestUser1",
   "repository": "SSMTestDocsRepo",
   "path": "SSMDocs/mySSMdoc.yml",
   "branch": "myBranch",
   "tokenInfo": "{\{ssm-secure:myAccessTokenParam\}}"
}

**Note**

"branch" is required only if your SSM document is stored in a branch other than master.
To use the version of your SSM document in a particular commit in your repository, use commitID with getOptions instead of branch. For example:

"getOptions": "commitID:bbc1ddb94...b76d3bEXAMPLE",

- If you choose S3, specify **Source Info** information in the following format:

  
  
  "path": "URL_to_document_in_S3"

  
  For example:

  
  
  {"path": "https://s3.amazonaws.com/aws-executecommand-test/scripts/ruby/mySSMdoc.json"}

- If you choose SSMDocument, specify **Source Info** information in the following format:

  
  
  {"name": "document_name"}

  
  For example:

  
  
  {"name": "mySSMdoc"}

6. In the **Document Parameters** field, type parameters for the remote SSM document. For example, if you run the AWS-RunPowerShell document, you could specify:

  
  
  {"commands": ["date", "echo \"Hello World\""]}

If you run the AWS-ConfigureAWSPack document, you could specify:

  
  
  {"action": "Install",
   "name": "AWSPVDriver"}

7. In the **Targets** section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

**Note**

If you choose to select instances manually, and an instance you expect to see is not included in the list, see [Where are my instances? (p. 862)](#) for troubleshooting tips.
8. For **Other parameters**:
   - For **Comment**, type information about this command.
   - For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

9. (Optional) For **Rate control**:
   - For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.
     
     **Note**
     If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
   
   - For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

10. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.
     
     **Note**
     The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see [Create an IAM instance profile for Systems Manager](p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

11. In the **SNS notifications** section, if you want notifications sent about the status of the command execution, select the **Enable SNS notifications** check box.

    For more information about configuring Amazon SNS notifications for Run Command, see [Monitoring Systems Manager status changes using Amazon SNS notifications](p. 1151).

12. Choose **Run**.

    **Note**
    For information about rebooting servers and instances when using Run Command to call scripts, see [Rebooting managed instance from scripts](p. 843).
Security in AWS Systems Manager

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Systems Manager, see AWS Services in Scope by Compliance Program.

- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company’s requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Systems Manager. The following topics show you how to configure Systems Manager to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Systems Manager resources.

**Topics**

- Data protection in AWS Systems Manager (p. 1102)
- Identity and access management for AWS Systems Manager (p. 1105)
- Using service-linked roles for Systems Manager (p. 1122)
- Logging and monitoring in AWS Systems Manager (p. 1127)
- Compliance validation for AWS Systems Manager (p. 1129)
- Resilience in AWS Systems Manager (p. 1129)
- Infrastructure security in AWS Systems Manager (p. 1130)
- Configuration and vulnerability analysis in AWS Systems Manager (p. 1130)
- Security best practices for Systems Manager (p. 1130)

### Data protection in AWS Systems Manager

AWS Systems Manager conforms to the AWS shared responsibility model, which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

Data protection refers to protecting data while *in transit* (as it travels to and from Systems Manager) and *at rest* (while it is stored in AWS data centers).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that each user is given only
the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a Name field. This includes when you work with Systems Manager or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Systems Manager or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

Data encryption

Encryption at rest

Parameter Store Parameters

The types of parameters you can create in Parameter Store include String, StringList, and SecureString.

To encrypt SecureString parameter values, Parameter Store uses an AWS Key Management Service (AWS KMS) customer master key (CMK). AWS KMS uses either a customer managed CMK or an AWS managed CMK to encrypt the parameter value in an AWS managed database.

**Important**

Do not store sensitive data in a String or StringList parameter. For all sensitive data that must remain encrypted, use only the SecureString parameter type.

For more information, see Parameter types and examples (p. 199) and SecureString parameters (p. 206).

Content in Amazon S3 Buckets

As part of your Systems Manager operations, you might choose to upload or store data in one or more Amazon Simple Storage Service (Amazon S3) buckets.

For information about S3 bucket encryption, see Protecting Data Using Encryption and Data Protection in Amazon S3 in the Amazon Simple Storage Service Developer Guide.

The following are types of data you can upload or have stored in S3 buckets as part of your Systems Manager activities.

- The output of Run Command commands
- Distributor packages
- Patch Manager patching operation logs
- Patch Manager patch override lists
- Scripts or Ansible Playbooks to run in an Automation document workflow
- Chef InSpec profiles for use with Compliance scans
• AWS CloudTrail logs
• Session Manager session history logs
• Reports from Explorer or OpsData from OpsCenter
• AWS CloudFormation templates for use with Automation workflows
• Compliance data from a Resource Data Sync scan
• Output of requests to create or edit State Manager association on managed instances
• Custom SSM documents that you can run using the AWS managed AWS-RunDocument document

CloudWatch Logs Log Groups

As part of your Systems Manager operations, you might choose to stream data to one or more Amazon CloudWatch Logs log groups.

For information about CloudWatch Logs log group encryption, see Encrypt Log Data in CloudWatch Logs in the Amazon CloudWatch Logs User Guide.

The following are types of data you might have streamed to a CloudWatch Logs log group as part of your Systems Manager activities.

• The output of Run Command commands
• The output of scripts run using the aws:executeScript action in an Automation document
• Session Manager session history logs
• Logs from SSM Agent on your managed instances

Encryption in transit

We recommend that you use an encryption protocol such as Transport Layer Security (TLS) to encrypt sensitive data in transit between clients and your instances.

Systems Manager provides the following support for encryption of your data in transit.

Connections to Systems Manager API endpoints

Systems Manager API endpoints only support secure connections over HTTPS. When you manage Systems Manager resources with the AWS Management Console, AWS SDK, or the Systems Manager API, all communication is encrypted with Transport Layer Security (TLS). For a full list of API endpoints, see Regions and Endpoints in the Amazon Web Services General Reference.

Managed instances

AWS provides secure and private connectivity between EC2 instances. In addition, we automatically encrypt in-transit traffic between supported instances in the same VPC or in peered VPCs, using AEAD algorithms with 256-bit encryption. This encryption feature uses the offload capabilities of the underlying hardware, and there is no impact on network performance. The supported instances are: C5n, G4, I3en, M5dn, M5n, P3dn, R5dn, and R5n.

Session Manager sessions

By default, Session Manager uses TLS 1.2 to encrypt session data transmitted between the local machines of users in your account and your EC2 instances. You can also choose to further encrypt the data in transit using a customer master key (CMK) that has been created in AWS Key Management Service.

Run Command access

By default, remote access to your instances using Run Command is encrypted using TLS 1.2, and requests to create a connection are signed using SigV4.
Internetwork traffic privacy

You can use Amazon Virtual Private Cloud (Amazon VPC) to create boundaries between resources in your managed instances and control traffic between them, your on-premises network, and the internet. For details, see (Optional) Create a Virtual Private Cloud endpoint (p. 37).

For more information about Amazon Virtual Private Cloud security, see Security in the Amazon VPC User Guide.

Identity and access management for AWS Systems Manager

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Systems Manager resources. IAM is an AWS service that you can use with no additional charge.

Topics
- Audience (p. 1105)
- Authenticating with identities (p. 1105)
- Managing access using policies (p. 1107)
- How AWS Systems Manager works with IAM (p. 1109)
- AWS Systems Manager identity-based policy examples (p. 1115)
- Troubleshooting AWS Systems Manager identity and access (p. 1120)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Systems Manager.

Service user – If you use the Systems Manager service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Systems Manager features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Systems Manager, see Troubleshooting AWS Systems Manager identity and access (p. 1120).

Service administrator – If you’re in charge of Systems Manager resources at your company, you probably have full access to Systems Manager. It’s your job to determine which Systems Manager features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Systems Manager, see How AWS Systems Manager works with IAM (p. 1109).

IAM administrator – If you’re an IAM administrator, you might want to learn details about how you can write policies to manage access to Systems Manager. To view example Systems Manager identity-based policies that you can use in IAM, see AWS Systems Manager identity-based policy examples (p. 1115).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see The IAM Console and Sign-in Page in the IAM User Guide.
You must be authenticated (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don’t use AWS tools, you must sign the request yourself. Do this using Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 Signing Process in the AWS General Reference.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using Multi-Factor Authentication (MFA) in AWS in the IAM User Guide.

AWS Account Root User

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An IAM user is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing Access Keys for IAM Users in the IAM User Guide. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An IAM group is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named IAMAdmins and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to Create an IAM User (Instead of a Role) in the IAM User Guide.

IAM roles

An IAM role is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM Roles in the IAM User Guide.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON Policies in the IAM User Guide.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which
resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM Policies in the IAM User Guide.

Identity-based policies can be further categorized as inline policies or managed policies. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing Between Managed Policies and Inline Policies in the IAM User Guide.

For information about AWS managed policies for Systems Manager, see AWS managed policies for AWS Systems Manager (p. 1114).

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

Access control lists (ACLs)

Access control lists (ACLs) are a type of policy that controls which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format. Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access Control List (ACL) Overview in the Amazon Simple Storage Service Developer Guide.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the Principal field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions Boundaries for IAM Entities in the IAM User Guide.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs Work in the AWS Organizations User Guide.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session Policies in the IAM User Guide.
Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy Evaluation Logic in the IAM User Guide.

How AWS Systems Manager works with IAM

Before you use IAM to manage access to Systems Manager, you should understand what IAM features are available to use with Systems Manager. To get a high-level view of how Systems Manager and other AWS services work with IAM, see AWS Services That Work with IAM in the IAM User Guide.

Topics

- Systems Manager identity-based policies (p. 1109)
- Systems Manager resource-based policies (p. 1112)
- Authorization based on Systems Manager tags (p. 1112)
- Systems Manager IAM roles (p. 1113)

Systems Manager identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Systems Manager supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the IAM User Guide.

Actions

The Action element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Systems Manager use the following prefix before the action: ssm:. For example, to grant someone permission to create a Systems Manager parameter with the Systems Manager PutParameter API operation, you include the ssm:PutParameter action in their policy. Policy statements must include either an Action or NotAction element. Systems Manager defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [
    "ssm:action1",
    "ssm:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action:

```
"Action": "ssm:Describe*"
```

To see a list of Systems Manager actions, see Actions Defined by AWS Systems Manager in the IAM User Guide.
Resources

The Resource element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

For example, the Systems Manager maintenance window resource has the following ARN format.

```
arn:aws:ssm:region:account-id:maintenancewindow/window-id
```

To specify the mw-0c50858d01EXAMPLE maintenance windows in your statement, use an ARN similar to the following.

```
"Resource": "arn:aws:ssm:us-east-2:123456789012:maintenancewindow/mw-0c50858d01EXAMPLE"
```

To specify all maintenance windows that belong to a specific account, use the wildcard (*).

```
```

For Parameter Store API actions, you can provide or restrict access to all parameters in one level of a hierarchy by using hierarchical names and AWS Identity and Access Management (IAM) policies as follows.

```
```

Some Systems Manager actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": 
```

Some Systems Manager API actions accept multiple resources. To specify multiple resources in a single statement, separate their ARNs with commas as follows.

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

**Note**

Most AWS services treat a colon (:) or a forward slash (/) as the same character in ARNs. However, Systems Manager requires an exact match in resource patterns and rules. When creating event patterns, be sure to use the correct ARN characters so that they match the resource’s ARN.

The following table describes the ARN formats for the resource types supported by Systems Manager.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>ARN format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>arn:aws:ssm:region:account-id:association/association-id</td>
</tr>
<tr>
<td>Resource type</td>
<td>ARN format</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Maintenance window</td>
<td>arn:aws:ssm:region:account-id:maintenancewindow/window-id</td>
</tr>
<tr>
<td>Managed instance inventory</td>
<td>arn:aws:ssm:region:account-id:managed-instance-inventory/managed-instance-id</td>
</tr>
<tr>
<td>Parameter</td>
<td>A one-level parameter:</td>
</tr>
<tr>
<td></td>
<td>• arn:aws:ssm:region:account-id:parameter/parameter-name/</td>
</tr>
<tr>
<td></td>
<td>A parameter named with a hierarchical construction:</td>
</tr>
<tr>
<td>Patch baseline</td>
<td>arn:aws:ssm:region:account-id:patchbaseline/patch-baseline-id</td>
</tr>
<tr>
<td>Session</td>
<td>arn:aws:ssm:region:account-id:session/session-id</td>
</tr>
<tr>
<td>All Systems Manager resources</td>
<td>arn:aws:ssm:*</td>
</tr>
<tr>
<td>All Systems Manager resources owned</td>
<td>arn:aws:ssm:region:account-id:*</td>
</tr>
<tr>
<td>by the specified account in the</td>
<td>specified Region</td>
</tr>
<tr>
<td>specified Region</td>
<td></td>
</tr>
</tbody>
</table>

1. For automation definitions, Systems Manager supports a second-level resource, version ID. In AWS, these second-level resources are known as subresources. Specifying a version subresource for an automation definition resource lets you provide access to certain versions of an automation definition. For example, you might want to ensure that only the latest version of an automation definition is used in your instance management.

2. To organize and manage parameters, you can create names for parameters with a hierarchical construction. With hierarchical construction, a parameter name can include a path that you define by using forward slashes. You can name a parameter resource with a maximum of fifteen levels. We suggest that you create hierarchies that reflect an existing hierarchical structure in your environment. For more information, see Creating Systems Manager parameters (p. 249).

3. In most cases, the session ID is constructed using the ID of the account user who started the session, plus an alphanumeric suffix. For example:

   arn:aws:us-east-2:111122223333:session/JohnDoe-1a2b3c4sEXAMPLE

   However, if the user ID is not available, the ARN is constructed this way instead:

   arn:aws:us-east-2:111122223333:session/session-1a2b3c4sEXAMPLE

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces in the Amazon Web Services General Reference.
For a list of Systems Manager resource types and their ARNs, see Resources Defined by AWS Systems Manager in the IAM User Guide. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS Systems Manager.

**Condition keys**

The Condition element (or Condition block) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM Policy Elements: Variables and Tags in the IAM User Guide.

Systems Manager defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the IAM User Guide.

To see a list of Systems Manager condition keys, see Condition Keys for AWS Systems Manager in the IAM User Guide. To learn with which actions and resources you can use a condition key, see Actions Defined by AWS Systems Manager.

For information about using the ssm:resourceTag/* condition key, see the following topics:

- Restrict access to root-level commands through SSM Agent (p. 92)
- Restricting Run Command access based on instance tags (p. 831)
- Restrict session access based on instance tags (p. 791)

For information about using the ssm:Recursive and ssm:Overwrite condition keys, see Organizing parameters into hierarchies (p. 240).

**Examples**

To view examples of Systems Manager identity-based policies, see AWS Systems Manager identity-based policy examples (p. 1115).

**Systems Manager resource-based policies**

Other AWS services, such as Amazon Simple Storage Service, support resource-based permissions policies. For example, you can attach a permissions policy to an S3 bucket to manage access permissions to that bucket.

Systems Manager does not support resource-based policies.

**Authorization based on Systems Manager tags**

You can attach tags to Systems Manager resources or pass tags in a request to Systems Manager. To control access based on tags, you provide tag information in the condition element of a policy using the ssm:resourceTag/key-name, aws:ResourceTag/key-name, aws:RequestTag/key-name,
or `aws:TagKeys` condition keys. You can add tags to the following resource types when you create or update them:

- Document
- Managed instance
- Maintenance window
- Parameter
- Patch baseline
- OpsItem

For information about tagging Systems Manager resources, see Tagging Systems Manager resources (p. 1163).

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Viewing Systems Manager documents based on tags (p. 1120).

**Systems Manager IAM roles**

An **IAM role** is an entity within your AWS account that has specific permissions.

**Using temporary credentials with Systems Manager**

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Systems Manager supports using temporary credentials.

**Service-linked roles**

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Systems Manager supports service-linked roles. For details about creating or managing Systems Manager service-linked roles, see Using service-linked roles for Systems Manager (p. 1122).

**Service roles**

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Systems Manager supports service roles.

**Choosing an IAM role in Systems Manager**

For Systems Manager to interact with your managed instances, you must choose a role to allow Systems Manager to access instances on your behalf. If you have previously created a service role or service-linked role, then Systems Manager provides you with a list of roles to choose from. It's important to choose a role that allows access to start and stop managed instances.

To access EC2 instances, the role your AWS account needs is an IAM instance profile. For information, see Create an IAM instance profile for Systems Manager (p. 30).
To access on-premises instances or virtual machines (VMs), the role your AWS account needs is an IAM service role for a hybrid environment. For information, see Create an IAM service role for a hybrid environment (p. 44).

An Automation workflow can be initiated under the context of a service role (or assume role). This allows the service to perform actions on your behalf. If you do not specify an assume role, Automation uses the context of the user who invoked the execution. However, certain situations require that you specify a service role for Automation. For more information, see Configuring a service role (assume role) access for Automation workflows (p. 281).

AWS managed policies for AWS Systems Manager

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. These AWS managed policies grant necessary permissions for common use cases so you can avoid having to investigate which permissions are needed. (You can also create your own custom IAM policies to allow permissions for Systems Manager actions and resources.) For more information, see AWS Managed Policies in the IAM User Guide.

The following AWS managed policies, which you can attach to users in your account, are specific to AWS Systems Manager:

- **AmazonSSMFullAccess** – User trust policy that grants full access to the Systems Manager API and documents.
- **AmazonSSMReadOnlyAccess** – User trust policy that grants access to Systems Manager read-only API actions, such as Get* and List*.
- **AmazonSSMAutomationApproverAccess** – User trust policy that enables access to view automation executions and send approval decisions to automation that is waiting for approval.
- **AmazonSSMAutomationRole** – Service role policy that provides permissions for the AWS Systems Manager automation service to run activities defined within automation documents. Assign this policy to administrators and trusted power users.
- **AmazonSSMMaintenanceWindowRole** – Service role policy for Systems Manager Maintenance Windows.
- **AmazonSSMDirectoryServiceAccess** – Instance trust policy that allows SSM Agent to access AWS Directory Service on behalf of the user for requests to join the domain by the managed instance.
- **AmazonSSMManagedInstanceCore** – Instance trust policy that enables an instance to use AWS Systems Manager service core functionality.
- **AmazonSSMServiceRolePolicy** – Service role policy that provides access to AWS resources managed or used by AWS Systems Manager.
- **AWSResourceAccessManagerServiceRolePolicy** – Service role policy containing read-only AWS Resource Access Manager access to the account's AWS Organizations structure. It also contains IAM permissions to self-delete the role.
- **AmazonEC2RoleforSSM** – This policy will be deprecated soon. In its place, use the AmazonSSMManagedInstanceCore policy to enable AWS Systems Manager service core functionality on EC2 instances. For information, see Create an IAM instance profile for Systems Manager (p. 30).

**Note**

In a hybrid environment, you need an additional IAM role that allows servers and VMs to communicate with the Systems Manager service. This is the IAM service role for Systems Manager. This role grants AWS Security Token Service (AWS STS) AssumeRole trust to the Systems Manager service. The AssumeRole action returns a set of temporary security credentials (consisting of an access key ID, a secret access key, and a security token). You use these temporary credentials to access AWS resources that you might not normally have access to. For more information, see Create an IAM service role for a hybrid environment (p. 44) and AssumeRole in AWS Security Token Service API Reference.
AWS Systems Manager identity-based policy examples

By default, IAM users and roles don't have permission to create or modify Systems Manager resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

The following is an example of a permissions policy that allows a user to delete documents with names that begin with \textit{MyDocument} in the \textit{us-west-2} Region.

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ssm:DeleteDocument"],
    }
  ]
}

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the IAM User Guide.

Topics
- Policy best practices (p. 1115)
- Using the Systems Manager console (p. 1116)
- Allow users to view their own permissions (p. 1116)
- Customer managed policy examples (p. 1117)
- Viewing Systems Manager documents based on tags (p. 1120)

Policy best practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Systems Manager resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- \textbf{Get Started Using AWS Managed Policies} – To start using Systems Manager quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get Started Using Permissions With AWS Managed Policies in the IAM User Guide.

- \textbf{Grant Least Privilege} – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant Least Privilege in the IAM User Guide.

- \textbf{Enable MFA for Sensitive Operations} – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using Multi-Factor Authentication (MFA) in AWS in the IAM User Guide.
• **Use Policy Conditions for Extra Security** – To the extent that it’s practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON Policy Elements: Condition in the IAM User Guide.

**Using the Systems Manager console**

To access the AWS Systems Manager console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Systems Manager resources and other resources in your AWS account.

To fully use Systems Manager in the Systems Manager console, you must have permissions from the following services:

- AWS Systems Manager
- Amazon Elastic Compute Cloud (Amazon EC2)
- AWS Identity and Access Management (IAM)

You can grant the required permissions with the following policy statement.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:*",
                "ec2:describeInstances",
                "iam:PassRole",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won’t function as intended for entities (IAM users or roles) with that policy.

You don’t need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you’re trying to perform.

**Allow users to view their own permissions**

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:GetUserPolicy"  
            ],
            "Resource": "*"
        }
    ]
}
```
Customer managed policy examples

You can create standalone policies that you administer in your own AWS account. We refer to these as customer managed policies. You can attach these policies to multiple principal entities in your AWS account. When you attach a policy to a principal entity, you give the entity the permissions that are defined in the policy. For more information, see Customer Managed Policies in IAM User Guide.

The following examples of user policies grant permissions for various AWS Systems Manager actions. Use them to limit the Systems Manager access for your IAM users and roles. These policies work when performing actions in the Systems Manager API, AWS SDKs, or the AWS CLI. For users who use the console, you need to grant additional permissions specific to the console. For more information, see Using the Systems Manager console (p. 1116).

**Note**

All examples use the US West (Oregon) Region (us-west-2) and contain fictitious account IDs.

**Examples**

- Example 1: Allow a user to perform Systems Manager operations in a single Region (p. 1117)
- Example 2: Allow a user to list documents for a single Region (p. 1118)

**Example 1: Allow a user to perform Systems Manager operations in a single Region**

The following example grants permissions to perform AWS Systems Manager operations only in the us-west-2 Region:

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
        "ssm:*",
        "iam:GetUser",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
        ],
        "Resource": ["arn:aws:iam::*:user/${aws:username}"],
    },
    {
        "Sid": "NavigateInConsole",
        "Effect": "Allow",
        "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:GetPolicy",
        "iam:GetUser",
        "iam:ListUsers"
        ],
        "Resource": "*
    }
    ]
}
```
Example 2: Allow a user to list documents for a single Region

The following example grants permissions to list all document names that begin with Update in the us-west-2 Region:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ssm:ListDocuments"],
        }
    ]
}
```

Example 3: Allow a user to use a specific SSM document to run commands on specific instances

The following example IAM policy allows a user to do the following.

- List Systems Manager documents and document versions.
- View details about documents.
- Send a command using the document specified in the policy. The name of the document is determined by this entry:

```
```

- Send a command to three instances. The instances are determined by the following entries in the second Resource section:

```
"arn:aws:ec2:us-east-2:aws-account-ID:instance/i-07782c72faEXAMPLE"
```

- View details about a command after it has been sent.
- Start and stop Automation executions.
- Get information about Automation executions.

If you want to give a user permission to use this document to send commands on any instance for which the user currently has access (as determined by their AWS user account), you could specify the following entry in the Resource section and remove the other instance entries.

```
"arn:aws:ec2:us-east-2:*,*,instance/**
```
"Version": "2012-10-17",
"Statement": [ 
    { "Action": [ "ssm:ListDocuments",
                  "ssm:ListDocumentVersions",
                  "ssm:DescribeDocument",
                  "ssm:GetDocument",
                  "ssm:DescribeInstanceInformation",
                  "ssm:DescribeDocumentParameters",
                  "ssm:DescribeInstanceProperties"
                ],
    "Effect": "Allow",
    "Resource": "*"
    },
    { "Action": "ssm:SendCommand",
      "Effect": "Allow",
                    "arn:aws:ec2:us-east-2:aws-account-ID:instance/i-07782c72faEXAMPLE",
                  ]
    },
    { "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/"
                  ]
    }
]
Viewing Systems Manager documents based on tags

You can use conditions in your identity-based policy to control access to Systems Manager resources based on tags. This example shows how you might create a policy that allows viewing an SSM document. However, permission is granted only if the document tag `Owner` has the value of that user's user name. This policy also grants the permissions necessary to complete this action on the console.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListDocumentsInConsole",
      "Effect": "Allow",
      "Action": "ssm:ListDocuments",
      "Resource": "*"
    },
    {
      "Sid": "ViewDocumentIfOwner",
      "Effect": "Allow",
      "Action": "ssm:GetDocument",
      "Resource": "arn:aws:ssm:*:*:document/*",
      "Condition": {
        "StringEquals": {"ssm:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

You can attach this policy to the IAM users in your account. If a user named `richard-roe` attempts to view a Systems Manager document, the document must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise, he is denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case-sensitive. For more information, see IAM JSON Policy Elements: Condition in the IAM User Guide.

Troubleshooting AWS Systems Manager identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Systems Manager and IAM.

**Topics**

- I Am not authorized to perform an action in Systems Manager (p. 1120)
- I Am not authorized to perform `iam:PassRole` (p. 1121)
- I want to view my access keys (p. 1121)
- I'm an administrator and want to allow others to access Systems Manager (p. 1121)
- I want to allow people outside of my AWS account to access my Systems Manager resources (p. 1122)

I Am not authorized to perform an action in Systems Manager

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.
The following example error occurs when the mateojackson IAM user tries to use the console to view details about a document but does not have ssm:GetDocument permissions.

```
User: arn:aws:ssm::123456789012:user/mateojackson is not authorized to perform: ssm:GetDocument on resource: MyExampleDocument
```

In this case, Mateo asks his administrator to update his policies to allow him to access the MyExampleDocument resource using the ssm:GetDocument action.

### I Am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the iam:PassRole action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Systems Manager.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named marymajor tries to use the console to perform an action in Systems Manager. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary asks her administrator to update her policies to allow her to perform the iam:PassRole action.

### I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, AKIAIOSFODNN7EXAMPLE) and a secret access key (for example, wJalrXUtFlnFEMI/K7MDENG/bPXRf1CYEXAMPLEKEY). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

**Important**

Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing Access Keys in the IAM User Guide.

### I'm an administrator and want to allow others to access Systems Manager

To allow others to access Systems Manager, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Systems Manager.
To get started right away, see Creating Your First IAM Delegated User and Group in the IAM User Guide.

I want to allow people outside of my AWS account to access my Systems Manager resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Systems Manager supports these features, see How AWS Systems Manager works with IAM (p. 1109).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing Access to an IAM User in Another AWS Account That You Own in the IAM User Guide.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing Access to AWS Accounts Owned by Third Parties in the IAM User Guide.
- To learn how to provide access through identity federation, see Providing Access to Externally Authenticated Users (Identity Federation) in the IAM User Guide.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM Roles Differ from Resource-based Policies in the IAM User Guide.

Using service-linked roles for Systems Manager

AWS Systems Manager uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Systems Manager. Service-linked roles are predefined by Systems Manager and include all the permissions that the service requires to call other AWS services on your behalf.

Topics

- Using Roles to Collect Inventory and Run Maintenance Windows Tasks (p. 1122)
- Using Roles to Collect AWS Account Information for Systems Manager Explorer (p. 1125)

Using Roles to Collect Inventory and Run Maintenance Windows Tasks

AWS Systems Manager uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Systems Manager. Service-linked roles are predefined by Systems Manager and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Systems Manager easier because you don't have to manually add the necessary permissions. Systems Manager defines the permissions of its service-linked roles, and unless defined otherwise, only Systems Manager can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy can't be attached to any other IAM entity.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.
Service-linked role permissions for Systems Manager

Systems Manager uses the service-linked role named **AWSServiceRoleForAmazonSSM** – AWS Systems Manager uses this IAM service role to manage AWS resources on your behalf.

The AWSServiceRoleForAmazonSSM service-linked role trusts only ssm.amazonaws.com to assume this role.

Currently, only two Systems Manager capabilities use the service-linked role:

- Inventory requires a service-linked role. The role enables the system to collect Inventory metadata from tags and Resource Groups.
- The Maintenance Windows capability can optionally use the service-linked role. The role enables the Maintenance Windows service to run maintenance tasks on target instances. Note that the service-linked role for Systems Manager doesn't provide the permissions needed for all scenarios. For more information, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622)

The AWSServiceRoleForAmazonSSM service-linked role permissions policy allows Systems Manager to complete the following actions on all related resources ("Resource": "*") except where indicated:

- ssm:CancelCommand
- ssm:GetCommandInvocation
- ssm:ListCommandInvocations
- ssm:ListCommands
- ssm:SendCommand
- ssm:GetAutomationExecution
- ssm:GetParameters
- ssm:StartAutomationExecution
- ssm:EventNotification
- ssm:SendAutomationEvent
- ssm:GetAutomationParameter
- ssm:DescribeAutomationParameters
- ssm:GetAutomationTargetsForParameter
- ssm:DescribeAutomationTargets
- ssm:GetAutomationPatchBaseline
- ssm:GetAutomationSummaries
- ssm:GetAutomationPatchBaselineSummaries
- ec2:DescribeInstanceAttribute
- ec2:DescribeInstanceStatus
- ec2:DescribeInstances
- lambda:InvokeFunction¹
- states:DescribeExecution²
- states:StartExecution²
- resource-groups:ListGroupResources
- resource-groups:GetGroupQuery
- cloudformation:DescribeStacks
- config:SelectResourceConfig
- compute-optimizer:GetEC2InstanceRecommendations
- support:DescribeTrustedAdvisorChecks
- support:DescribeTrustedAdvisorCheckSummaries
- support:DescribeTrustedAdvisorCheckResult
- iam:PassRole³

¹ The lambda:InvokeFunction action is allowed permissions for the following resources only:

```
arn:aws:lambda:*:*:function:SSM*
```
Creating a service-linked role for Systems Manager

You can use the IAM console to create a service-linked role with the EC2 use case. In the IAM CLI or the IAM API, create a service-linked role with the ssm.amazonaws.com service name. For more information, see Creating a Service-Linked Role in the IAM User Guide.

For maintenance windows only, you don't need to manually create a service-linked role. When you create a maintenance window task in the AWS Management Console, the AWS CLI, or the Systems Manager API, Systems Manager creates the service-linked role for you if you choose not to provide a custom service role.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account.

Editing a service-linked role for Systems Manager

Systems Manager does not allow you to edit the AWSServiceRoleForAmazonSSM service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the IAM User Guide.

Deleting a service-linked role for Systems Manager

If you no longer need to use any feature or service that requires a service-linked role, then we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. You can use the IAM console, the IAM CLI, or the IAM API to manually delete the service-linked role. To do this, you must first manually clean up the resources for your service-linked role, and then you can manually delete it.

Because the Systems Manager service-linked role can be used by both the Inventory and Maintenance Windows capabilities, ensure that neither is using the role before attempting to delete it.

- **Inventory**: If you delete the service-linked role used by Systems Manager Inventory, then the Inventory data for tags and Resource Groups will no longer be synchronized. You must clean up the resources for your service-linked role before you can manually delete it.
- **Maintenance Windows**: You can't delete the service-linked role if any maintenance window tasks currently rely on the role. You must first remove the service-linked role from the tasks before you can delete the role.
Note
If the Systems Manager service is using the role when you try to delete the tags, Resource Groups, or maintenance window tasks, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete Systems Manager resources used by the AWSServiceRoleForAmazonSSM
1. To delete tags, see Adding and Deleting Tags on an Individual Resource.
2. To delete Resource Groups, see Delete Groups from AWS Resource Groups.
3. For information about how to delete maintenance window tasks, see Update or delete maintenance window tasks (console) (p. 640).

To manually delete the service-linked role using IAM
Use the IAM console, the IAM CLI, or the IAM API to delete the AWSServiceRoleForAmazonSSM service-linked role. For more information, see Deleting a Service-Linked Role in the IAM User Guide.

Supported regions for Systems Manager service-linked roles
Systems Manager supports using service-linked roles in all of the regions where the service is available. For more information, see AWS Regions and Endpoints for Systems Manager.

Using Roles to Collect AWS Account Information for Systems Manager Explorer

AWS Systems Manager uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Systems Manager. Service-linked roles are predefined by Systems Manager and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up Systems Manager easier because you don't have to manually add the necessary permissions. Systems Manager defines the permissions of its service-linked roles, and unless defined otherwise, only Systems Manager can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your Systems Manager resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Systems Manager

Systems Manager uses the service-linked role named AWSServiceRoleForAmazonSSM_AccountDiscovery – AWS Systems Manager uses this IAM service role to call other AWS services to discover AWS account information.

The AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role trusts the following services to assume the role:

* accountdiscovery.ssm.amazonaws.com

The role permissions policy allows Systems Manager to complete the following actions on the specified resources:
• organizations:DescribeAccount
• organizations:DescribeOrganization
• organizations:ListAccounts
• organizations:ListAWSServiceAccessForOrganization
• organizations:ListChildren
• organizations:ListParents

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-Linked Role Permissions in the IAM User Guide.

Creating a service-linked role for Systems Manager

You don't need to manually create a service-linked role. When you create a resource data sync by using Systems Manager Explorer in the AWS Management Console, the AWS CLI, or the AWS API, Systems Manager creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create a resource data sync by using Systems Manager Explorer, Systems Manager creates the service-linked role for you again.

Editing a service-linked role for Systems Manager

Systems Manager does not allow you to edit the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a Service-Linked Role in the IAM User Guide.

Deleting a service-linked role for Systems Manager

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can manually delete it.

Cleaning up a service-linked role

Before you can use IAM to delete a service-linked role, you must first delete all Explorer resource data syncs. For more information, see Deleting a Systems Manager Explorer Resource Data Sync (p. 134).

Note
If the Systems Manager service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

Manually delete the service-linked role

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForAmazonSSM_AccountDiscovery service-linked role. For more information, see Deleting a Service-Linked Role in the IAM User Guide.

Supported Regions for Systems Manager service-linked roles

Systems Manager supports using service-linked roles in all of the regions where the service is available. For more information, see AWS Regions and Endpoints.

Systems Manager does not support using service-linked roles in every region where the service is available. You can use the AWSServiceRoleForAmazonSSM_AccountDiscovery role in the following regions.
Logging and monitoring in AWS Systems Manager

Monitoring is an important part of maintaining the reliability, availability, and performance of Systems Manager and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Systems Manager and other resources and responding to potential incidents.

**AWS CloudTrail Logs**

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Systems Manager. Using the information collected by CloudTrail, you can determine the request that was made to Systems Manager, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144).

**Amazon CloudWatch Alarms**

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify for your EC2 instances and other resources. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions because they are in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Using Amazon CloudWatch Alarms in the Amazon CloudWatch User Guide.
Amazon CloudWatch Dashboards

CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources. Amazon CloudWatch dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different Regions. You can use CloudWatch dashboards to create customized views of the metrics and alarms for your AWS resources. For more information, see Amazon CloudWatch dashboards hosted by Systems Manager (p. 168).

Amazon CloudWatch Events

CloudWatch Events lets you configure rules to alert you to changes in Systems Manager resources, and to direct CloudWatch Events to take actions based on the content of those events. CloudWatch Events provides support for a number of events that are emitted by various Systems Manager capabilities. For more information, see Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147).

Amazon CloudWatch Logs and SSM Agent Logs

SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each instance. You can view log files by manually connecting to an instance. We recommend automatically sending agent log data to a log group in CloudWatch Logs for analysis. For more information, see Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135) and View SSM Agent logs (p. 90).

AWS Systems Manager Configuration Compliance

You can use AWS Systems Manager Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren't compliant. By default, Configuration Compliance displays current compliance data about Systems Manager Patch Manager patching and Systems Manager State Manager associations. For more information, see AWS Systems Manager Configuration Compliance (p. 696).

AWS Systems Manager Explorer

Explorer is a customizable operations dashboard that reports information about your AWS resources. Explorer displays an aggregated view of operations data (OpsData) for your AWS accounts and across Regions. In Explorer, OpsData includes metadata about your EC2 instances, patch compliance details, and operational work items (OpsItems). Explorer provides context about how OpsItems are distributed across your business units or applications, how they trend over time, and how they vary by category. You can group and filter information in Explorer to focus on items that are relevant to you and that require action. For more information, see AWS Systems Manager Explorer (p. 118).

AWS Systems Manager OpsCenter

OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. OpsCenter aggregates and standardizes OpsItems across services while providing contextual investigation data about each OpsItem, related OpsItems, and related resources. OpsCenter also provides Systems Manager Automation documents that you can use to quickly resolve issues. OpsCenter is integrated with Amazon CloudWatch Events. This means you can create CloudWatch Events rules that automatically create OpsItems for any AWS service that publishes events to CloudWatch Events. For more information, see AWS Systems Manager OpsCenter (p. 138).

Amazon Simple Notification Service

You can configure Amazon Simple Notification Service (Amazon SNS) to send notifications about the status of commands that you send using Systems Manager Run Command or Maintenance Windows. Amazon SNS coordinates and manages sending and delivering notifications to clients or endpoints that are subscribed to Amazon SNS topics. You can receive a notification whenever a command
Compliance validation for AWS Systems Manager

This topic addresses AWS Systems Manager compliance with third-party assurance programs. For information about viewing compliance data for your managed instances, see AWS Systems Manager Configuration Compliance (p. 696).

Third-party auditors assess the security and compliance of AWS Systems Manager as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Systems Manager is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- **Security and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- **Architecting for HIPAA Security and Compliance Whitepaper** – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **Evaluating Resources with Rules** in the AWS Config Developer Guide – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in AWS Systems Manager

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability
Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

**Infrastructure security in AWS Systems Manager**

As a managed service, AWS Systems Manager is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Systems Manager through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

**Configuration and vulnerability analysis in AWS Systems Manager**

AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see the following resources:

- Compliance validation for AWS Systems Manager (p. 1129)
- Shared Responsibility Model
- Amazon Web Services: Overview of Security Processes (whitepaper)

**Security best practices for Systems Manager**

Systems Manager provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

**Topics**

- Systems Manager preventative security best practices (p. 1130)
- Systems Manager monitoring and auditing best practices (p. 1132)

**Systems Manager preventative security best practices**

The following best practices for Systems Manager can help prevent security incidents.

**Implement least privilege access**

When granting permissions, you decide who is getting what permissions to which Systems Manager resources. You enable specific actions that you want to allow on those resources. Therefore you should grant only the permissions that are required to perform a task. Implementing least privilege
access is fundamental in reducing security risk and the impact that could result from errors or malicious intent.

The following tools are available to implement least privilege access:
- IAM user policies and Permissions Boundaries for IAM Entities
- Service Control Policies

**Use SecureString parameters to encrypt and protect secret data**

A SecureString parameter is any sensitive data that needs to be stored and referenced in a secure manner. If you have data that you don't want users to alter or reference in plaintext, such as passwords or license keys, create those parameters using the SecureString datatype. Parameter Store uses an AWS Key Management Service (AWS KMS) customer master key (CMK) to encrypt the parameter value. AWS KMS uses either a customer managed CMK or an AWS managed CMK when encrypting the parameter value. For maximum security, we recommend using your own CMK. If you use the default AWS managed CMK key, any user with permission to run the GetParameter and GetParameters actions in your account can view or retrieve the content of all SecureString parameters. If you are using customer managed CMKs to encrypt your secure SecureString values, you can use IAM policies and key policies to manage permissions for encrypting and decrypting parameters. You can't establish access control policies for these operations when you use the default AWS managed CMK.

For more information, see SecureString parameters (p. 206) and How AWS Systems Manager Parameter Store Uses AWS KMS in the AWS Key Management Service Developer Guide.

**Define allowedValues and allowedPattern for document parameters**

You can validate user input for SSM document parameters by defining allowedValues and allowedPattern. For allowedValues, you define an array of values allowed for the parameter. If a user inputs a value that is not allowed, the execution fails to start. For allowedPattern, you define a regular expression that validates whether the user input matches the defined pattern for the parameter. If the user input does not match the allowed pattern, the execution fails to start.

For more information about allowedValues and allowedPattern, see SSM document syntax (p. 1046).

**Use an Amazon Virtual Private Cloud (Amazon VPC) and VPC endpoints**

You can use Amazon VPC to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

By implementing a VPC endpoint, you can privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

For more information about Amazon VPC security, see (Optional) Create a Virtual Private Cloud endpoint (p. 37) and Security in the Amazon VPC User Guide.

**Restrict Session Manager users to sessions using interactive commands**

AWS Systems Manager Session Manager provides several methods for starting sessions (p. 818) to your managed instances. For the most secure connections, you can require users to connect using the interactive commands method to limit user interaction to a specific command or command sequence. This helps you manage the interactive actions a user can take. For more information, see Starting a session (interactive commands) (p. 821).

**Provide temporary instance permissions for Automation executions**

During an Automation execution, your instances might need permissions that are needed for that execution only but not for other Systems Manager operations. For example, an Automation
Workflow might require an instance to call an particular API action or access an AWS resource specifically during the workflow. If these calls or resources are ones that you want to limit access to, you can provide temporary, supplemental permissions for your instances within the Automation document itself instead of adding the permissions to your IAM instance profile. At the end of the Automation execution, the temporary permissions are removed. For more information, see Providing temporary instance permissions with AWS Systems Manager Automations on the AWS Management and Governance Blog.

Keep AWS and Systems Manager tools up to date

AWS regularly releases updated versions of tools and plugins that you can use in your AWS and Systems Manager operations. Keeping these resources up to date ensures that users and instances in your account have access to the latest functionality and security features in these tools.

- **SSM Agent** – AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure these resources. We recommend checking for new versions, or automating updates to the agent, at least every two weeks. For information, see Automate updates to SSM Agent (p. 93).

- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands in your command-line shell. To update the AWS CLI, you run the same command used to install the AWS CLI. We recommend creating a scheduled task on your local machine to run the command appropriate to your operating system at least once every two weeks. For information about installation commands, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide.

- **AWS Tools for Windows PowerShell** – The Tools for Windows PowerShell are a set of PowerShell modules that are built on the functionality exposed by the AWS SDK for .NET. The AWS Tools for Windows PowerShell enable you to script operations on your AWS resources from the PowerShell command line. Periodically, as updated versions of the AWS Tools for Windows PowerShell are released, you should update the version that you are running locally. For information, see Updating the AWS Tools for Windows PowerShell on Windows or Updating the AWS Tools for Windows PowerShell on Linux or macOS.

- **Session Manager plugin** – If users in your organization with permissions to use Session Manager want to connect to an instance using the AWS CLI, they must first install the Session Manager plugin on their local machines. To update the plugin, you run the same command used to install the plugin. We recommend creating a scheduled task on your local machine to run the command appropriate to your operating system at least once every two weeks. For information, see (Optional) Install the Session Manager Plugin for the AWS CLI (p. 813).

- **CloudWatch agent** – You can configure and use the CloudWatch agent to collect metrics and logs from your EC2 instances, on-premises instances, and virtual machines (VMs). These logs can be sent to Amazon CloudWatch Logs for monitoring and analysis. We recommend checking for new versions, or automating updates to the agent, at least every two weeks. For the simplest updates, use AWS Systems Manager Quick Setup. For information, see AWS Systems Manager Quick Setup (p. 16).

Systems Manager monitoring and auditing best practices

The following best practices for Systems Manager can help detect potential security weaknesses and incidents.

**Identify and audit all your Systems Manager resources**

Identification of your IT assets is a crucial aspect of governance and security. You need to identify all of your Systems Manager resources to assess their security posture and take action on potential areas of weakness.
Use Tag Editor to identify security-sensitive or audit-sensitive resources, then use those tags when you need to search for these resources. For more information, see Searching for Resources to Tag in the AWS Resource Groups User Guide.

Create resource groups for your Systems Manager resources. For more information, see What Is AWS Resource Groups?

**Implement monitoring using Amazon CloudWatch monitoring tools**

Monitoring is an important part of maintaining the reliability, security, availability, and performance of Systems Manager and your AWS solutions. CloudWatch provides several tools and services to help you monitor Systems Manager and your other AWS services. For more information, see Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135) and Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147).

**Use AWS CloudTrail**

AWS CloudTrail provides a record of actions taken by a user, role, or an AWS service in Systems Manager. Using the information collected by AWS CloudTrail, you can determine the request that was made to Systems Manager, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144).

**Enable AWS Config**

AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config monitors resource configurations, allowing you to evaluate the recorded configurations against the desired secure configurations. Using AWS Config, you can review changes in configurations and relationships between AWS resources, investigate detailed resource configuration histories, and determine your overall compliance against the configurations specified in your internal guidelines. This can help you simplify compliance auditing, security analysis, change management, and operational troubleshooting. For more information, see Setting Up AWS Config with the Console in the AWS Config Developer Guide. When specifying the resource types to record, ensure that you include Systems Manager resources.

**Monitor AWS security advisories**

You should regularly check security advisories posted in Trusted Advisor for your AWS account. You can do this programmatically using describe-trusted-advisor-checks.

Further, actively monitor the primary email address registered to each of your AWS accounts. AWS will contact you, using this email address, about emerging security issues that might affect you.

AWS operational issues with broad impact are posted on the AWS Service Health Dashboard. Operational issues are also posted to individual accounts via the Personal Health Dashboard. For more information, see the AWS Health Documentation.

**Related Content**

Amazon Web Services: Overview of Security Processes (whitepaper)


IAM Best Practices

Security Best Practices in AWS CloudTrail

Security Best Practices for Amazon S3
Monitoring AWS Systems Manager

Monitoring is an important part of maintaining the reliability, availability, and performance of Systems Manager and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs. But before you start monitoring Systems Manager, you should create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

After you have defined your monitoring goals and have created your monitoring plan, the next step is to establish a baseline for normal Systems Manager performance in your environment. You should measure Systems Manager performance at various times and under different load conditions. As you monitor Systems Manager, you should store a history of monitoring data that you've collected. You can compare current Systems Manager performance to this historical data to help you to identify normal performance patterns and performance anomalies, and devise methods to address them.

For example, you can monitor the success or failure of operations such as Automation workflows, the application of patch baselines, maintenance window events, and configuration compliance.

You can also monitor CPU utilization, disk I/O, and network utilization of your managed instances. When performance falls outside your established baseline, you might need to reconfigure or optimize the instance to reduce CPU utilization, improve disk I/O, or reduce network traffic. For more information about monitoring EC2 instances, see Monitoring Amazon EC2 in the Amazon EC2 User Guide for Linux Instances.

Topics
- Monitoring tools (p. 1134)
- Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135)
- Sending instance logs to CloudWatch Logs (SSM Agent) (p. 1141)
- Monitoring Run Command metrics using Amazon CloudWatch (p. 1142)
- Logging AWS Systems Manager API calls with AWS CloudTrail (p. 1144)
- Configuring Amazon CloudWatch Logs for Run Command (p. 1146)
- Monitoring Systems Manager events with Amazon CloudWatch Events (p. 1147)
- Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151)

Monitoring tools

The content in this chapter provides information for using tools available for monitoring your Systems Manager and other AWS resources. For a more complete list of tools, see Logging and monitoring in AWS Systems Manager (p. 1127).
Sending instance logs to CloudWatch Logs (CloudWatch agent)

You can configure and use the Amazon CloudWatch agent to collect metrics and logs from your instances instead of using SSM Agent for these tasks. The CloudWatch agent enables you to gather more metrics on EC2 instances than are available using SSM Agent. In addition, you can gather metrics from on-premises servers using the CloudWatch agent.

You can also store agent configuration settings in the Systems Manager Parameter Store for use with the CloudWatch agent.

**Note**
Currently, AWS Systems Manager supports migrating from SSM Agent to the CloudWatch agent for collecting logs and metrics on 64-bit versions of Windows only. For information about setting up the CloudWatch agent on other operating systems, and for complete information about using the CloudWatch agent, see Collect metrics from Amazon Elastic Compute Cloud instances and on-premises servers with the CloudWatch agent in the Amazon CloudWatch User Guide.

You can use the CloudWatch agent on other supported operating systems, but you will not be able to use Systems Manager to perform a tool migration.

SSM Agent writes information about executions, scheduled actions, errors, and health statuses to log files on each instance. Manually connecting to an instance to view log files and troubleshoot an issue with SSM Agent is time-consuming. For more efficient instance monitoring, you can configure either SSM Agent itself or the CloudWatch agent to send this log data to Amazon CloudWatch Logs.

**Important**
The unified CloudWatch agent has replaced SSM Agent as the tool for sending log data to Amazon CloudWatch Logs. Support for using SSM Agent to send log data will be deprecated in the near future. We recommend using only the unified CloudWatch agent for your log collection processes. For more information, see the following topics:

- Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135)
- Migrate Windows Server instance log collection to the CloudWatch agent (p. 1136)
- Collect metrics from Amazon Elastic Compute Cloud instances and on-premises servers with the CloudWatch agent in the Amazon CloudWatch User Guide

Using CloudWatch Logs, you can monitor log data in real-time, search and filter log data by creating one or more metric filters, and archive and retrieve historical data when you need it. For more information about CloudWatch Logs, see the Amazon CloudWatch Logs User Guide.

Configuring an agent to send log data to Amazon CloudWatch Logs provides the following benefits:

- Centralized log file storage for all of your SSM Agent log files.
- Quicker access to files to investigate errors.
- Indefinite log file retention (configurable).
- Logs can be maintained and accessed regardless of the status of the instance.
- Access to other CloudWatch features such as metrics and alarms.

For information about monitoring Session Manager activity, see Auditing and logging session activity (p. 823).
Migrate Windows Server instance log collection to the CloudWatch agent

If you are currently using SSM Agent on supported Windows Server instances to send SSM Agent log files to Amazon CloudWatch Logs, you can use Systems Manager to migrate from SSM Agent to the CloudWatch agent as your log collection tool, as well as migrate your configuration settings.

The CloudWatch agent is not supported on 32-bit versions of Windows Server.

For 64-bit EC2 instances for Windows Server, you can perform the migration to the CloudWatch agent automatically or manually. For on-premises servers and virtual machines, the process must be performed manually.

**Note**
During the migration process, the data sent to CloudWatch may be interrupted or duplicated. Your metrics and log data will be recorded accurately again in CloudWatch after the migration is completed.

We recommend testing the migration on a limited number of instances before migrating an entire fleet to the CloudWatch agent. After migration, if you prefer log collection with SSM Agent, you can return to using it instead.

**Important**
In the following cases, you won’t be able to migrate to the CloudWatch agent using the steps described in this topic:

- The existing configuration for SSM Agent specifies multiple Regions.
- The existing configuration for SSM Agent specifies multiple sets of access/secret key credentials.

In these cases, it will be necessary to disable log collection in SSM Agent and install the CloudWatch agent without a migration process. For more information, see the following topics:

- Install the CloudWatch agent on an EC2 instance
- Install the CloudWatch agent on an on-premises server

**Before You Begin**

Before you begin a migration to the CloudWatch agent for log collection, ensure that the instances on which you will perform the migration meet these requirements:

- The OS is a 64-bit version of Windows Server.
- SSM Agent 2.2.93.0 or later is installed on the instance.
- SSM Agent is configured for monitoring on the instance.

**Topics**

- Automatically migrating to the CloudWatch agent (p. 1136)
- Manually migrating to the CloudWatch agent (p. 1138)

**Automatically migrating to the CloudWatch agent**

For EC2 instances for Windows Server only, you can use the AWS Systems Manager console or the AWS CLI to automatically migrate to the CloudWatch agent as your log collection tool.
Note
Currently, AWS Systems Manager supports migrating from SSM Agent to the CloudWatch agent for collecting logs and metrics on 64-bit versions of Windows only. For information about setting up the CloudWatch agent on other operating systems, and for complete information about using the CloudWatch agent, see Collect metrics from Amazon Elastic Compute Cloud instances and on-premises servers with the CloudWatch agent in the Amazon CloudWatch User Guide.
You can use the CloudWatch agent on other supported operating systems, but you will not be able to use Systems Manager to perform a tool migration.

After the migration succeeds, check your results in CloudWatch to ensure you are receiving the metrics, logs, or Windows event logs you expect. If you are satisfied with the results, you can optionally Store CloudWatch agent configuration settings in Parameter Store (p. 1139). If the migration is not successful or the results are not as expected, you can Rolling back to log collection with SSM Agent (p. 1140).

Note
If you want to migrate a source configuration file that includes a {hostname} entry, then be aware that the {hostname} entry can change the value of the field after the migration is complete. For example, say that the following "LogStream": "{hostname}" entry maps to a server named MyLogServer001.

```json
{
  "Id": "CloudWatchIISLogs",
  "Parameters": {
    "AccessKey": "",
    "SecretKey": "",
    "Region": "us-east-1",
    "LogGroup": "Production-Windows-IIS",
    "LogStream": "{hostname}"
  }
}
```

After the migration, this entry will map to a domain, such as ip-11-1-1-11.production.ExampleCompany.com. To retain the local hostname value, specify {local_hostname} instead of {hostname}.

To automatically migrate to the CloudWatch agent (console)
2. In the navigation pane, choose Run Command, and then choose Run command.
3. In the Command document list, choose AmazonCloudWatch-MigrateCloudWatchAgent.
4. In the Targets section, choose an option and select the instances to update.
5. Choose Run.

To automatically migrate to the CloudWatch agent (AWS CLI)
- Run the following command:

```
aws ssm send-command --document-name AmazonCloudWatch-MigrateCloudWatchAgent --targets Key=instanceids,Values=ID1,ID2,ID3
```

ID1, ID2, and ID3 represent the IDs of instances you want to update, such as i-02573cafEXAMPLE.
Manually migrating to the CloudWatch agent

For on-premises Windows Server instances or EC2 instances for Windows Server, follow these steps to manually migrate log collection to the Amazon CloudWatch agent.

Note
If you want to migrate a source configuration file that includes a `hostname` entry, then be aware that the `hostname` entry can change the value of the field after the migration is complete. For example, say that the following "LogStream": "{hostname}" entry maps to a server named MyLogServer001.

```json
{
  "Id": "CloudWatchIISLogs",
  "Parameters": {
    "AccessKey": "",
    "SecretKey": "",
    "Region": "us-east-1",
    "LogGroup": "Production-Windows-IIS",
    "LogStream": "{hostname}"
  }
}
```

After the migration, this entry will map to a domain, such as ip-11-1-1-11.production.ExampleCompany.com. To retain the local hostname value, specify `{local_hostname}` instead of `{hostname}`.

One: To install the CloudWatch agent (console)

2. In the navigation pane, choose Run Command, and then choose Run command.
3. In the Command document list, choose AWS-ConfigureAWSPackage.
4. In the Targets section, choose an option and select the instances to update.
5. In the Action list, choose Install.
6. In Name, type AmazonCloudWatchAgent.
7. In Version, type latest if it is not already provided by default.

Two: To update config data JSON format

- To update the JSON formatting of the existing config settings for the CloudWatch agent, use AWS Systems Manager Run Command or log into the instance directly with an RDP connection to run the following Windows PowerShell commands on the instance, one at a time:

```powershell
cd \{Env:ProgramFiles\}\Amazon\AmazonCloudWatchAgent
\{Env:ProgramFiles\}\amazon-cloudwatch-agent-config-wizard.exe --isNonInteractiveWindowsMigration
```

{Env:ProgramFiles} represents the location where the Amazon folder containing the CloudWatch agent can be found, typically C:\Program Files.

Three: To configure and start the CloudWatch agent (console)

2. In the navigation pane, choose Run Command, and then choose Run command.
3. In the Command document list, choose AWS-RunPowerShellScript.
4. In the Targets section, choose an option and select the instances to update.
5. In the Commands box, enter the following two commands:

   ```bash
   cd "${Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent-ctl.ps1 -a fetch-config -m ec2 -c file:config.json -s
   ```

   `{Env:ProgramFiles}` represents the location where the Amazon folder containing the CloudWatch agent can be found, typically C:\Program Files.

Four: To disable log collection in SSM Agent (console)

2. In the navigation pane, choose Run Command, and then choose Run command.
3. In the Command document list, choose AWS-ConfigurecloudWatch.
4. In the Targets section, choose an option and select the instances to update.
5. In the Status list, choose Disabled.

After completing these steps, check your logs in CloudWatch to ensure you are receiving the metrics, logs, or Windows event logs you expect. If you are satisfied with the results, you can optionally Store CloudWatch agent configuration settings in Parameter Store (p. 1139). If the migration is not successful or the results are not as expected, you can Rolling back to log collection with SSM Agent (p. 1140).

Store CloudWatch agent configuration settings in Parameter Store

You can store the contents of an Amazon CloudWatch agent configuration file in Parameter Store. By maintaining this configuration data in a parameter, multiple instances can derive their configuration settings from it, and you avoid having to create or manually update configuration files on your instances. For example, you can use Run Command to write the contents of the parameter to configuration files on multiple instances, or use State Manager to help avoid configuration drift in the CloudWatch agent configuration settings across a fleet of instances.

When you run the CloudWatch agent configuration wizard, you can choose to let the wizard save your configuration settings as a new parameter in Parameter Store. For information about running the CloudWatch agent configuration wizard, see Create the CloudWatch agent configuration file with the wizard.

If you ran the wizard but did not choose the option to save the settings as a parameter, or you created the CloudWatch agent configuration file manually, you can retrieve the data to save as a parameter on your instance in the following file:

```bash
#{Env:ProgramFiles}\Amazon\AmazonCloudWatchAgent\config.json
```

`{Env:ProgramFiles}` represents the location where the Amazon folder containing the CloudWatch agent can be found, typically C:\Program Files.
Rolling back to log collection with SSM Agent

If you want to return to using SSM Agent for log collection, follow these steps.

One: To retrieve config data from SSM Agent

1. On the instance where you want to return to collecting logs with the SSM Agent, locate the contents of the SSM Agent config file. This JSON file is typically found in the following location:

```bash
${Env:ProgramFiles}\Amazon\SSM\Plugins\awsCloudWatch\AWS.EC2.Windows.CloudWatch.json
```

{Env:ProgramFiles} represents the location where the Amazon folder can be found, typically C:\Program Files.

2. Copy this data into a text file for use in a later step.

   We recommend storing a backup of the JSON on a location other than the instance itself.

Two: To uninstall the CloudWatch agent (console)

2. In the navigation pane, choose Run Command, and then choose Run command.
3. In the Command document list, choose AWS-ConfigureAWSPackage.
4. In the Targets section, choose an option and select the instances to update.
5. In the Action list, choose Uninstall.
6. In Name, type AmazonCloudWatchAgent.
7. Choose Run.

Three: To reenable log collection in SSM Agent (console)

2. In the navigation pane, choose Run Command, and then choose Run command.
3. In the Command document list, choose AWS-ConfigureCloudWatch.
4. In the Targets section, choose an option and select the instances to update.
5. In the Status list, choose Enabled.
6. In the Properties box, paste the contents of the old config data you saved to the text file.
7. Choose Run.
Sending instance logs to CloudWatch Logs (SSM Agent)

AWS Systems Manager Agent (SSM Agent) is Amazon software that runs on your EC2 instances and your hybrid instances (on-premises instances and virtual machines) that are configured for Systems Manager. SSM Agent processes requests from the Systems Manager service in the cloud and configures your machine as specified in the request. For more information about SSM Agent, see Working with SSM Agent (p. 63).

In addition, following the steps below, you can configure SSM Agent to send log data to Amazon CloudWatch Logs.

**Important**
The unified CloudWatch agent has replaced SSM Agent as the tool for sending log data to Amazon CloudWatch Logs. Support for using SSM Agent to send log data will be deprecated in the near future. We recommend using only the unified CloudWatch agent for your log collection processes. For more information, see the following topics:

- Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135)
- Migrate Windows Server instance log collection to the CloudWatch agent (p. 1136)
- Collect metrics from Amazon Elastic Compute Cloud instances and on-premises servers with the CloudWatch agent in the Amazon CloudWatch User Guide

**Before You Begin**
Create a log group in Amazon CloudWatch Logs. For more information, see Create a log group in CloudWatch Logs in the Amazon CloudWatch Logs User Guide.

**To configure SSM Agent to send logs to CloudWatch**

1. Log into an instance and locate the following file:

   **Linux**
   `/etc/amazon/ssm/seelog.xml.template`

   **Windows**
   `%ProgramFiles%\Amazon\SSM\seelog.xml.template`

2. Change the file name from `seelog.xml.template` to `seelog.xml`.

3. Open the `seelog.xml` file in a text editor, and locate the following section.

   **Linux**
   ```xml
   <outputs formatid="fmtinfo">
   <console formatid="fmtinfo"/>
   <rollingfile type="size" filename="/var/log/amazon/ssm/amazon-ssm-agent.log" maxsize="30000000" maxrolls="5"/>
   <filter levels="error,critical" formatid="fmterror">
   <rollingfile type="size" filename="/var/log/amazon/ssm/errors.log" maxsize="10000000" maxrolls="5"/>
   </filter>
   </outputs>
   ```

1141
4. Edit the file, and add a custom name element after the closing </filter> tag. In the following example, the custom name as been specified as cloudwatch_receiver.

Linux

```
<outputs formatid="fmtinfo">
  <console formatid="fmtinfo"/>
  <rollingfile type="size" filename="/var/log/amazon/ssm/amazon-ssm-agent.log" maxsize="30000000" maxrolls="5"/>
  <filter formatid="fmterror" levels="error,critical">
    <rollingfile type="size" filename="/var/log/amazon/ssm/errors.log" maxsize="10000000" maxrolls="5"/>
  </filter>
  <custom name="cloudwatch_receiver" formatid="fmtdebug" data-log-group="your-CloudWatch-log-group-name"/>
</outputs>
```

Windows

```
<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. Save your changes, and then restart SSM Agent or the instance.
7. In the navigation pane, choose Log groups, and then choose the name of your log group.

Tip
The log stream for SSM Agent log file data is organized by instance ID.

Monitoring Run Command metrics using Amazon CloudWatch

**Metrics** are the fundamental concept in CloudWatch. A metric represents a time-ordered set of data points that are published to CloudWatch. Think of a metric as a variable to monitor, and the data points as representing the values of that variable over time.
AWS Systems Manager publishes metrics about the status of Run Command commands to CloudWatch, enabling you to set alarms based on those metrics. These statistics are recorded for an extended period so you can access historical information and gain a better perspective on the success rate of commands run in your AWS account.

The terminal status values for commands for which you can track metrics include Success, Failed, and Delivery Timed Out. For example, for an SSM command document set to run every hour, you can configure an alarm to notify you when a status of Success is not reported for any of those hours. For more information about command status values, see Understanding command statuses (p. 845).

**To view metrics in the CloudWatch console**

2. In the navigation pane, choose Metrics.
3. Select the **SSM Run Command** tile.

**To view metrics using the AWS CLI**

Open a command prompt, and use the following command:

```bash
aws cloudwatch list-metrics --namespace "AWS/SSM-RunCommand"
```

To list all available metrics, use the following command:

```bash
aws cloudwatch list-metrics --namespace "AWS/SSM-RunCommand"
```

---

**Systems Manager Run Command metrics and dimensions**

Systems Manager sends Systems Manager Run Command command metrics to CloudWatch one time every minute.

Systems Manager sends the following command metrics to CloudWatch.

**Note**

All of these metrics use Count as the unit, so Sum and SampleCount are the most useful statistics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommandsDeliveryTimedOut</td>
<td>The number of commands that have a terminal status of Delivery Timed Out.</td>
</tr>
<tr>
<td>CommandsFailed</td>
<td>The number of commands that have a terminal status of Failed.</td>
</tr>
<tr>
<td>CommandsSucceeded</td>
<td>The number of commands that have a terminal status of Success.</td>
</tr>
</tbody>
</table>

For more information about working with CloudWatch metrics, see the following topics in the Amazon CloudWatch User Guide:

- Metrics
Logging AWS Systems Manager API calls with AWS CloudTrail

Systems Manager is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Systems Manager. CloudTrail captures all API calls for Systems Manager as events, including calls from the Systems Manager console and from code calls to the Systems Manager APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an S3 bucket, including events for Systems Manager. If you don’t configure a trail, you can still view the most recent events in the CloudTrail console in Event history. Using the information collected by CloudTrail, you can determine the request that was made to Systems Manager, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Systems Manager information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Systems Manager, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Systems Manager, create a trail. A trail enables CloudTrail to deliver log files to an S3 bucket. By default, when you create a trail in the console, the trail applies to all regions. The trail logs events from all regions in the AWS partition and delivers the log files to the S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All Systems Manager actions are logged by CloudTrail and are documented in the AWS Systems Manager API Reference. For example, calls to the CreateMaintenanceWindows, PutInventory, SendCommand, and StartSession actions generate entries in the CloudTrail log files. (For an example of setting up CloudTrail to monitor a Systems Manager API call, see Monitoring session activity using Amazon CloudWatch Events (console) (p. 826).)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.
Understanding Systems Manager log file entries

A trail is a configuration that enables delivery of events as log files to an S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the DeleteDocuments action on a document named example-Document in the US East (Ohio) Region (us-east-2).

```
{
  "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",
    "sourceIp": "203.0.113.11",
    "userAgent": "example-user-agent-string",
    "requestParameters": {
      "name": "example-Document"
    },
    "responseElements": null,
    "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
    "eventID": "832b082d5-d474-44e8-a51d-093ccEXAMPLE",
    "resources": [
      {
        "accountId": "123456789012"
      }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }
}
```
Configuring Amazon CloudWatch Logs for Run Command

When you send a command by using Run Command, you can specify where you want to send the command output. By default, Systems Manager returns only the first 2,500 characters of the command output. If you want to view the full details of the command output, you can specify an Amazon Simple Storage Service (Amazon S3) bucket. Or you can specify Amazon CloudWatch Logs. If you specify CloudWatch Logs, Run Command periodically sends all command output and error logs to CloudWatch Logs. You can monitor output logs in near real-time, search for specific phrases, values, or patterns, and create alarms based on the search.

If you configured your instance or on-premises hybrid machine to use the AWS Identity and Access Management (IAM) managed policies `AmazonSSMManagedInstanceCore` and `CloudWatchAgentServerPolicy`, then your instance requires no additional configuration to send output to CloudWatch Logs. You simply need to choose this option if sending commands from the console, or add the `cloud-watch-output-config` section and `CloudWatchOutputEnabled` parameter if using the AWS CLI, Tools for Windows PowerShell, or an API action. The `cloud-watch-output-config` section and `CloudWatchOutputEnabled` parameter are described in more detail later in this topic.

For information about adding policies to an instance profile for EC2 instances, see Create an IAM instance profile for Systems Manager (p. 30). For information about adding policies to a service role for on-premises instances and virtual machines that you plan to use as managed instances, see Create an IAM service role for a hybrid environment (p. 44).

For information about updating an existing instance profile, see Add permissions to a Systems Manager instance profile (console) (p. 34).

If you are using a custom policy on your instances, then you must update the policy on each instance to allow Systems Manager to send output and logs to CloudWatch Logs. Add the following policy objects to your custom policy. For more information, about updating an IAM policy, see Editing IAM policies in the IAM User Guide.

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

Specifying CloudWatch Logs when you send commands

To specify CloudWatch Logs as the output when you send a command from the AWS Management Console, choose CloudWatch Output in the Output options section. Optionally, you can specify the name of CloudWatch Logs group where you want to send command output. If you don't specify a group name, Systems Manager automatically creates a log group for you. The log group uses the following naming format: /aws/ssm/SystemsManagerDocumentName.

If you run commands by using the AWS CLI, then you must specify the `cloud-watch-output-config` section in your command. This section enables you to specify the `CloudWatchOutputEnabled` parameter, and optionally, the `CloudWatchLogGroupName` parameter. Here is an example:
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}'

Viewing command output in CloudWatch Logs

As soon as the command starts to run, Systems Manager sends output to CloudWatch Logs in near real-time. The output in CloudWatch Logs uses the following format:

```
CommandID/InstanceID/PluginID/stdout
CommandID/InstanceID/PluginID/stderr
```

Output from the execution is uploaded every 30 seconds or when the buffer exceeds 200 KB, whichever happens first.

**Note**

Log Streams are only created when output data is available. For example, if there is no error data for an execution, the stderr stream isn’t created.

Here is an example of the command output as it appears in CloudWatch Logs.

```
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
```

Monitoring Systems Manager events with Amazon CloudWatch Events

You can configure rules in Amazon CloudWatch Events to alert you to changes in Systems Manager resources, and to direct CloudWatch Events to take actions based on the content of those events. CloudWatch Events provides support for a number of events that are emitted by various Systems Manager capabilities.

**Note**

For Systems Manager actions that aren’t supported by CloudWatch Events, you can create an event rule that is based on an API call, which are recorded by AWS CloudTrail. For an example, see Monitoring session activity using Amazon CloudWatch Events (console) (p. 826).

For more information about how to get started with CloudWatch Events and set up rules, see Getting Started with CloudWatch Events in the Amazon CloudWatch Events User Guide.

Following are lists of the Systems Manager event types with built-in monitoring support in CloudWatch Events.

**Run Command**

Supported events include the following:

- Status change for a command (applies to one or more instances).
- Status change for a command invocation (applies to one instance only).

For more information, see Configuring CloudWatch Events for Run Command (p. 1149).

**Automation**

Supported events include the following:
• Status change for an automation execution.
• Status change for a single step in an automation execution.

For more information, see Configuring CloudWatch Events for Systems Manager automation (p. 1150).

State Manager

Supported events include the following:
• State change for an association
• State change for an instance association.

Configuration Compliance

Supported events include the following:
• State change for association compliance.
• State change for instance patch compliance.

For more information, see Remediating compliance issues (p. 702).

Maintenance Window

Supported events include the following:
• State change for a maintenance window (enabled or disabled)
• Change in a maintenance window target registration.
• Change in a maintenance window task registration.
• State change for a maintenance window execution.
• State change for a maintenance window task execution.
• State change for a maintenance window task target invocation.

Parameter Store

Supported events include the following:
• A parameter is created, updated, or deleted, or a label is attached or moved from one version to another (detail-type: "Parameter Store Change").
• A parameter has expired or been deleted, its expiration date is approaching, or its value hasn't been changed for a specified period of time (detail-type: "Parameter Store Policy Action").

For more information, see Setting up notifications or trigger actions based on Parameter Store events (p. 226).

Inventory

Supported events include the following:
• Deletion of custom inventory item on an instance.
• Availability of a delete action summary.
• A disabled custom inventory type is detected.

For more information, see Viewing inventory delete actions in CloudWatch Events (p. 744).

For more information about the Systems Manager event types that are supported by CloudWatch Events, see the following topics in the Amazon CloudWatch Events User Guide:

• AWS Systems Manager Events
• AWS Systems Manager Configuration Compliance Events
• AWS Systems Manager Maintenance Windows Events
• AWS Systems Manager Parameter Store Events
Configuring CloudWatch Events for Run Command

Use Amazon CloudWatch Events to log command execution status changes. You can create a rule that runs whenever there is a state transition, or when there is a transition to one or more states that are of interest.

You can also specify Run Command as a target action when a CloudWatch event occurs. For example, say a CloudWatch event is triggered that an instance in an Auto Scaling group is about to terminate. You can configure CloudWatch so the target of that event is a Run Command script that captures the log files from the instance before it is terminated. You can also configure a Run Command action when a new instance is created in an Auto Scaling group. For example, when CloudWatch receives the instance-created event, Run Command could enable the web server role or install software on the instance.

• Configuring CloudWatch Events for Run Command (p. 1149)
• Configure Run Command as a CloudWatch Events target (p. 1149)

Configuring CloudWatch Events for Run Command

You can configure CloudWatch Events to notify you of Run Command status changes, or a status change for a specific command invocation. Use the following procedure to configure CloudWatch Events to send notification about Run Command.

To configure CloudWatch Events for Run Command

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, under Events, choose Rules, and then choose Create rule.
3. Under Event Source, verify that Event Pattern is selected.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM)
5. In the Event Type field, choose Run Command.
6. Choose the detail types and statuses for which you want to receive notifications, and then choose Add targets.
7. In the Select target type list, choose a target type. For information about the different types of targets, see the corresponding AWS Help documentation.
8. Choose Configure details.
9. Specify the rule details, and then choose Create rule.

Configure Run Command as a CloudWatch Events target

Use the following procedure to configure a Run Command action as the target of a CloudWatch event.

To configure Run Command as a target of a CloudWatch event

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose Events, and then either choose to create a new rule or edit an existing rule.
3. After specifying or verifying the details of the rule, choose Add target.
4. In the Select target type list, choose SSM Run Command.
5. In the Document list, choose an SSM document. The document determines the type of actions Run Command can perform on your instances.
Note
Verify that the document you choose can run on the instance operating system. Some documents run only on Windows or only on Linux operating systems. For more information about SSM Documents, see AWS Systems Manager documents (p. 1031).

6. In the Target key field, specify either InstanceIds or tag:EC2_tag_name. Here are some examples of a Target key that uses an EC2 tag: tag:production and tag:server-role.

7. In the Target value(s) field, if you chose InstanceIds in the previous step, specify one or more instance IDs separated by commas. If you chose tag:EC2_tag_name in the previous step, specify one or more tag values. After you type the value, for example web-server or database, choose Add.

8. In the Configure parameter(s) section, choose an option and then complete any fields populated by your choice. Use the hover text for more information about the options. For more information about the parameter fields for your document, see Running commands using Systems Manager Run Command (p. 833) and choose the procedure for your document.

9. In the permissions section, choose Create a new role for this specific resource to create a new role with the required instance profile role for Run Command. Or, choose Use existing role. For more information about roles required for Run Command, see Setting up AWS Systems Manager (p. 25).

10. Choose Configure details and complete the wizard.

**Configuring CloudWatch Events for Systems Manager automation**

You can configure Amazon CloudWatch Events to notify you of Systems Manager Automation events. For example, you can configure CloudWatch Events to send notifications when an Automation step succeeds or fails. You can also configure CloudWatch Events to send notifications if the Automation workflow succeeds or fails.

**Tip**
For information about using the input transformer feature of Amazon CloudWatch Events to extract the instance-id of an EC2 instance from an instance state change event, see Walkthrough: Using input transformers with Automation (p. 601).

Use the following procedure to configure CloudWatch Events to send notification about Automation events.

**To configure CloudWatch Events for Automation**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. Choose Events in the left navigation, and then choose Create rule.
3. Under Event Source, verify that Event Pattern is selected.
4. In the Service Name field, choose EC2 Simple Systems Manager (SSM)
5. In the Event Type field, choose Automation.
6. Choose the detail types and statuses for which you want to receive notifications, and then choose Add targets.
7. In the Select target type list, choose a target type. For information about the different types of targets, see the corresponding AWS Help documentation.
8. Choose Configure details.
9. Specify the rule details, and then choose Create rule.

The next time you run Automation, CloudWatch Events sends event details to the target you specified.
Monitoring Systems Manager status changes using Amazon SNS notifications

You can configure Amazon Simple Notification Service (Amazon SNS) to send notifications about the status of commands that you send using AWS Systems Manager Run Command or AWS Systems Manager Maintenance Windows. Amazon SNS coordinates and manages sending and delivering notifications to clients or endpoints that are subscribed to Amazon SNS topics. You can receive a notification whenever a command changes to a new state or to a specific state, such as Failed or Timed Out. In cases where you send a command to multiple instances, you can receive a notification for each copy of the command sent to a specific instance. Each copy is called an invocation.

Amazon SNS can deliver notifications as HTTP or HTTPS POST, email (SMTP, either plaintext or in JSON format), or as a message posted to an Amazon Simple Queue Service (Amazon SQS) queue. For more information, see What is Amazon SNS in the Amazon Simple Notification Service Developer Guide. For examples of the structure of the JSON data included in the Amazon SNS notification provided by Run Command and Maintenance Windows, see Example Amazon SNS notifications for AWS Systems Manager (p. 1156).

Configure Amazon SNS notifications for AWS Systems Manager

Run Command and Run Command tasks that are registered to a maintenance window can send Amazon SNS notifications for command tasks that enter the following statuses. For information about the conditions that cause a command to enter one of these statuses, see Understanding command statuses (p. 845).

- In Progress
- Success
- Failed
- Timed Out
- Canceled

**Note**

Commands sent using Run Command also report Canceling and Pending status. These statuses are not captured by Amazon SNS notifications.

Command summary Amazon SNS notifications

If you configure Run Command or a Run Command task in your maintenance window for Amazon SNS notifications, Amazon SNS sends summary messages that include the following information.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventTime</td>
<td>String</td>
<td>The time that the event was triggered. The timestamp is important because Amazon SNS does not guarantee message delivery order. Example: 2016-04-26T13:15:30Z</td>
</tr>
<tr>
<td>documentName</td>
<td>String</td>
<td>The name of the SSM document used to run this command.</td>
</tr>
</tbody>
</table>
## Configure Amazon SNS notifications for AWS Systems Manager

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>commandId</td>
<td>String</td>
<td>The ID generated by Run Command after the command was sent.</td>
</tr>
<tr>
<td>expiresAfter</td>
<td>Date</td>
<td>If this time is reached and the command has not already started executing, it will not run.</td>
</tr>
<tr>
<td>outputS3BucketName</td>
<td>String</td>
<td>The Amazon Simple Storage Service (Amazon S3) bucket where the responses to the command execution should be stored.</td>
</tr>
<tr>
<td>outputS3KeyPrefix</td>
<td>String</td>
<td>The Amazon S3 directory path inside the bucket where the responses to the command execution should be stored.</td>
</tr>
<tr>
<td>requestedDateTime</td>
<td>String</td>
<td>The time and date that the request was sent to this specific instance.</td>
</tr>
<tr>
<td>instanceIds</td>
<td>StringList</td>
<td>The instances that were targeted by the command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Instance IDs are only included in the summary message if the Run Command task targeted instance IDs directly. Instance IDs are not included in the summary message if the Run Command task was issued using tag-based targeting.</td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>Command status for the command.</td>
</tr>
</tbody>
</table>

### Invocation-based Amazon SNS notifications

If you send a command to multiple instances, Amazon SNS can send messages about each copy or invocation of the command. The messages include the following information.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventTime</td>
<td>String</td>
<td>The time that the event was triggered. The timestamp is important because Amazon SNS does not guarantee message delivery order. Example: 2016-04-26T13:15:30Z</td>
</tr>
</tbody>
</table>
Configure Amazon SNS notifications for AWS Systems Manager

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>documentName</td>
<td>String</td>
<td>The name of the Systems Manager document used to run this command.</td>
</tr>
<tr>
<td>requestedDateTime</td>
<td>String</td>
<td>The time and date that the request was sent to this specific instance.</td>
</tr>
<tr>
<td>commandId</td>
<td>String</td>
<td>The ID generated by Run Command after the command was sent.</td>
</tr>
<tr>
<td>instanceId</td>
<td>String</td>
<td>The instance that was targeted by the command.</td>
</tr>
<tr>
<td>status</td>
<td>String</td>
<td>Command status for this invocation.</td>
</tr>
</tbody>
</table>

To set up Amazon SNS notifications when a command changes status, you must complete the following tasks.

**Note**
If you are not configuring Amazon SNS notifications for your maintenance window, then you can skip Task 5 below.

**Topics**
- Task 1: Create and subscribe to an Amazon SNS topic (p. 1153)
- Task 2: Create an IAM policy for Amazon SNS notifications (p. 1154)
- Task 3: Create an IAM role for Amazon SNS notifications (p. 1154)
- Task 4: Configure user access (p. 1155)
- Task 5: Attach the iam:PassRole policy to your maintenance window role (p. 1155)

**Task 1: Create and subscribe to an Amazon SNS topic**

An Amazon SNS topic is a communication channel that Run Command and Run Command tasks that are registered to a maintenance window use to send notifications about the status of your commands. Amazon SNS supports different communication protocols, including HTTP/S, email, and other AWS services like Amazon SQS. To get started quickly, we recommend that you start with the email protocol. For information about how to create a topic, see Create a topic in the Amazon Simple Notification Service Developer Guide.

**Note**
After you create the topic, copy or make a note of the **Topic ARN**. You specify this ARN when you send a command that is configured to return status notifications.

After you create the topic, subscribe to it by specifying an **Endpoint**. If you chose the Email protocol, the endpoint is the email address where you want to receive notifications. For more information about how to subscribe to a topic, see Subscribing an endpoint to an Amazon SNS topic in the Amazon Simple Notification Service Developer Guide.

Amazon SNS sends a confirmation email from **AWS Notifications** to the email address that you specify. Open the email and choose the **Confirm subscription** link.

You will receive an acknowledgement message from AWS. Amazon SNS is now configured to receive notifications and send the notification as an email to the email address that you specified.
Task 2: Create an IAM policy for Amazon SNS notifications

Use the following procedure to create a custom AWS Identity and Access Management (IAM) policy that provides permissions for triggering Amazon SNS notifications.

To create a custom IAM policy for Amazon SNS notifications

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies, and then choose Create policy. (If a Get Started button appears, choose it, and then choose Create Policy.)
3. Choose the JSON tab.
4. Replace the default content with the following:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "sns:Publish"
         ],
         "Resource": "sns-topic-arn"
      }
   ]
}
```

`sns-topic-arn` represents the ARN of the existing Amazon Simple Notification Service (Amazon SNS) topic to use to send Amazon Simple Notification Service notifications, in the format of `arn:aws:iam::account-id:sns-topic-name`. For example: `arn:aws:iam::123456789012:my-sns-topic`.

5. Choose Review policy.
6. On the Review policy page, for Name, enter a name for the inline policy. For example: `my-sns-publish-permissions`.
7. (Optional) For Description, enter a description for the policy.
8. Choose Create policy.

Task 3: Create an IAM role for Amazon SNS notifications

Use the following procedure to create an IAM role for Amazon SNS notifications. This service role is used by Systems Manager to trigger Amazon SNS notifications.

To create an IAM service role for Amazon SNS notifications

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles, and then choose Create role.
3. Under Select type of trusted entity, choose AWS service.
4. In the Choose a use case section, choose Systems Manager, and then choose Next: Permissions.
5. On the Attach permissions policies page, select the box to the left of the name of the custom policy you created in Task 2. For example: `my-sns-publish-permissions`.
6. Choose Next: Tags.
7. (Optional) Add one or more tag-key value pairs to organize, track, or control access for this role, and then choose Next: Review.
8. On the Review page, type a name in the Role name box, such as `my-sns-role`.
9. (Optional) Change the default role description to reflect the purpose of this role. For example: *Triggers SNS topics on your behalf.*
10. Choose Create role. The system returns you to the Roles page.
11. Copy or make a note of the Role ARN. This Role ARN is used when you send a command that is configured to return Amazon SNS notifications.
12. Leave the Summary page open.

**Task 4: Configure user access**

If your AWS Identity and Access Management (IAM) user account, group, or role is assigned administrator permissions, then you have access to Systems Manager Run Command and Maintenance Windows. If you don't have administrator permissions, then an administrator must give you permission by assigning the AmazonSSMFullAccess managed policy, or a policy that provides comparable permissions, to your IAM account, group, or role.

Use the following procedure to configure a user account to use Run Command and Maintenance Windows. If you need to create a new user account, see Creating an IAM user in your AWS account in the IAM User Guide.

**To configure user access and attach the iam:PassRole policy to a user account**

1. In the IAM navigation pane, choose Users, and then choose the user account that you want to configure.
2. On the Permissions tab, in the policies list, verify that either the AmazonSSMFullAccess policy is listed or that there is a comparable policy that gives the account permissions to access Systems Manager.
3. Choose Add inline policy.
4. On the Create policy page, choose the Visual editor tab.
5. Choose Service, and then choose IAM.
6. Choose Select actions.
7. In the Filter actions text box, type PassRole, and then choose the PassRole option.
8. Choose Resources. Verify that Specific is selected, and then choose Add ARN.
9. In the Specify ARN for role field, paste the Amazon SNS role ARN that you copied at the end of Task 3. The system automatically populates the Account and Role name with path fields.
10. Choose Add.
12. On the Review Policy page, type a name and then choose Create Policy.

**Task 5: Attach the iam:PassRole policy to your maintenance window role**

When you register a Run Command task with a maintenance window, you specify a service role Amazon Resource Name (ARN). This service role is used by Systems Manager to run tasks registered to the maintenance window. To configure Amazon SNS notifications for a registered Run Command task, you must attach an iam:PassRole policy to the maintenance window service role specified. If you do not intend to configure the registered task for Amazon SNS notifications, then this task can be skipped.

The iam:PassRole policy allows the Maintenance Windows service role to pass the SNS role created in Task 3 to the Amazon SNS service. The following procedure shows how to attach the iam:PassRole policy to the Maintenance Windows service role.
AWS Systems Manager User Guide

Example Amazon SNS notifications for AWS Systems Manager

**Note**
You must use a custom service role for your maintenance window to send notifications related to the Run Command tasks registered. For information, see Should I use a service-linked role or a custom service role to run maintenance window tasks? (p. 622).
If you need to create a custom service role, see one of the following topics:

- Control access to maintenance windows (console) (p. 622)
- Control access to maintenance windows (AWS CLI) (p. 626)
- Control access to maintenance windows (Tools for Windows PowerShell) (p. 630)

**To attach the iam:PassRole policy to your Maintenance Windows role**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose **Roles** and select the Amazon SNS role created in Task 3.
3. Copy or make a note of the **Role ARN** and return to the **Roles** section of the IAM console.
4. Select the custom Maintenance Windows service role you created (under **Role** name).
5. Under **Permissions**, verify that either the AmazonSSMMaintenanceWindowRole policy is listed or there is a comparable policy that gives maintenance windows permission to the Systems Manager API.
6. Choose **Add inline policy**.
7. Choose the **Visual Editor** tab.
8. For **Service**, choose IAM.
9. For **Actions**, choose PassRole.
10. For **Resources**, choose Specific, and then choose **Add ARN**.
11. In the **Specify ARN for role** box, paste the ARN of the Amazon SNS IAM role created in Task 3, and then choose **Add**.
12. Choose **Review policy**.
13. On the **Review Policy** page, specify a name for the PassRole policy, and then choose **Create policy**.

**Example Amazon SNS notifications for AWS Systems Manager**

You can configure Amazon Simple Notification Service (Amazon SNS) to send notifications about the status of commands that you send using AWS Systems Manager Run Command or AWS Systems Manager Maintenance Windows.

**Note**
This guide does not address how to configure notifications for Run Command or Maintenance Windows. For information about configuring Run Command or Maintenance Windows to send Amazon SNS notifications about the status of commands, see Configure Amazon SNS notifications for AWS Systems Manager (p. 1151).

The following examples show the structure of the JSON output returned by Amazon SNS notifications when configured for Run Command or Maintenance Windows.

**Sample JSON Output for Command summary messages using instance ID targeting**

```json
{
"commandId": "a8c7e76f-15f1-4c33-9052-0123456789ab",
"documentName": "AWS-RunPowerShellScript",
"instanceIds": [
"i-1234567890abcdef0",
```
Use Run Command to send a command that returns status notifications

The following procedures show how to use the AWS Command Line Interface (AWS CLI) or AWS Systems Manager console to send a Run Command that is configured to return status notifications.

Sending a Run Command that returns notifications (console)

Use the following procedure to send a command through Run Command that is configured to return status notifications using the Systems Manager console.

To send a command that returns notifications (console)

2. In the navigation pane, choose Run Command.

   -or-

   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Run Command.
3. Choose Run command.
4. In the Command document list, choose a Systems Manager document.
5. In the **Command parameters** section, specify values for required parameters.

6. In the **Targets** section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.

    **Note**
    If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.

7. For **Other parameters**:

    • For **Comment**, type information about this command.

    • For **Timeout (seconds)**, specify the number of seconds for the system to wait before failing the overall command execution.

8. (Optional) For **Rate control**:

    • For **Concurrency**, specify either a number or a percentage of instances on which to run the command at the same time.

        **Note**
        If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.

    • For **Error threshold**, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.

9. (Optional) For **Output options**, to save the command output to a file, select the **Write command output to an S3 bucket** box. Type the bucket and prefix (folder) names in the boxes.

    **Note**
    The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

10. In the **SNS Notifications** section, choose **Enable SNS notifications**.

11. In the **IAM role** field, type or paste the SNS IAM role ARN you created in Task 3 in the topic Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).

12. In the **SNS topic** field, type or paste the Amazon SNS topic ARN to be used.

13. In the **Notify me on** field, choose the events for which you want to receive notifications.

14. In the **Notify me for** field, choose to receive notifications for each copy of a command sent to multiple instances (invocations) or the command summary.

15. Choose **Run**.

16. Check your email for a message from Amazon SNS and open the email. Amazon SNS can take a few minutes to send the mail.

---

**Sending a Run Command that returns notifications (CLI)**

Use the following procedure to send a command through Run Command that is configured to return status notifications using the AWS CLI.

**To send a command that returns notifications (CLI)**

1. Open the AWS CLI.

2. Specify parameters in the following command to target based on managed instance IDs:
Use a maintenance window to send a command that returns status notifications

The following procedures show how to register an AWS Systems Manager Run Command task with your maintenance window using the Systems Manager console or the AWS Command Line Interface (AWS CLI). The procedures also describe how to configure the Run Command task to return status notifications.

Before You Begin

If you haven't created a maintenance window or registered targets, see Working with maintenance windows (console) (p. 634) for steps on how to create a maintenance window and register targets.

To receive notifications from the Amazon SNS service, you must attach an `iam:PassRole` policy to the Maintenance Windows service role specified in the registered task. If you haven't added `iam:PassRole` permissions to your Maintenance Windows service role, see Task 5: Attach the `iam:PassRole` policy to your maintenance window role (p. 1155).

Use a maintenance window to send a command that returns status notifications

aws ssm send-command --instance-ids "ID-1, ID-2" --document-name "Name"
--parameters "{"commands":["input"]}" --service-role "SNSRoleARN" --notification-config '
{"NotificationArn":"SNSTopicName","NotificationEvents":
["All"],"NotificationType":"Command"}'

For example:

```
aws ssm send-command --instance-ids "i-02573cafcfEXAMPLE, i-0471e04240EXAMPLE"
--document-name "AWS-RunPowerShellScript" --parameters "{"commands":
["Get-Process"]}" --service-role "arn:aws:iam::111122223333:role/
SNS_Role" --notification-config '
{"NotificationArn":"arn:aws:sns:us-east-1:111122223333:SNSTopic","NotificationEvents":
["All"],"NotificationType":"Command"}'
```

Alternative commands

Specify parameters in the following command to target managed instances using tags:

```
aws ssm send-command --targets "Key=tag:TagName,Values=TagKey" --document-name
"Name" --parameters "{"commands":["input"]}" --service-role "SNSRoleARN" --notification-config '
{"NotificationArn":"SNSTopicName","NotificationEvents":
["All"],"NotificationType":"Command"}'
```

For example:

```
aws ssm send-command --targets "Key=tag:Environment,Values=Dev" --document-name "AWS-RunPowerShellScript" --parameters "{"commands":
["Get-Process"]}" --service-role "arn:aws:iam::111122223333:role/
SNS_Role" --notification-config '
{"NotificationArn":"arn:aws:sns:us-east-1:111122223333:SNSTopic","NotificationEvents":
["All"],"NotificationType":"Command"}'
```

3. Press Enter.
4. Check your email for a message from Amazon SNS and open the email. Amazon SNS can take a few minutes to send the mail.

For more information about configuring Run Command from the command line, see Amazon EC2 Systems Manager API Reference and the Systems Manager AWS CLI Reference.
Registering a Run Command task to a maintenance window that returns notifications (console)

Use the following procedure to register a Run Command task that is configured to return status notifications to your maintenance window using the Systems Manager console.

**To register a Run Command task with your maintenance window that returns notifications (console)**

2. In the navigation pane, choose Maintenance Windows.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon to open the navigation pane, and then choose Maintenance Windows.
3. Select the maintenance window for which you would like to register a Run Command task configured to send Amazon SNS notifications.
4. Choose Actions and then choose Register Run Command task.
5. In the Name field, enter a name for the task.
6. In the Description field, enter a description.
7. From the Document list, choose a Command document.
8. In the Task priority list, specify a priority for this task. 1 is the highest priority. Tasks in a maintenance window are scheduled in priority order. Tasks that have the same priority are scheduled in parallel.
9. In the Targets section, identify the instances on which you want to run this operation by specifying tags, selecting instances manually, or specifying a resource group.
   **Note**  
   If you choose to select instances manually, and an instance you expect to see is not included in the list, see Where are my instances? (p. 862) for troubleshooting tips.
10. (Optional) For Rate control:
   - For Concurrency, specify either a number or a percentage of instances on which to run the command at the same time.
     **Note**  
     If you selected targets by specifying tags applied to managed instances or by specifying AWS resource groups, and you are not certain how many instances are targeted, then restrict the number of instances that can run the document at the same time by specifying a percentage.
   - For Error threshold, specify when to stop running the command on other instances after it fails on either a number or a percentage of instances. For example, if you specify three errors, then Systems Manager stops sending the command when the fourth error is received. Instances still processing the command might also send errors.
11. In the IAM service role area, choose the Maintenance Windows service role that has iam:PassRole permissions to the SNS role.
   **Note**  
   You must add iam:PassRole permissions to the Maintenance Windows role to enable Systems Manager to pass the SNS role to Amazon SNS. If you haven’t added iam:PassRole permissions, see Task 5 in the topic Monitoring Systems Manager status changes using Amazon SNS notifications (p. 1151).
12. (Optional) For Output options, to save the command output to a file, select the Write command output to an S3 bucket box. Type the bucket and prefix (folder) names in the boxes.
Use a maintenance window to send a command that returns status notifications

Note
The S3 permissions that grant the ability to write the data to an S3 bucket are those of the instance profile assigned to the instance, not those of the IAM user performing this task. For more information, see Create an IAM instance profile for Systems Manager (p. 30). In addition, if the specified S3 bucket is in a different AWS account, ensure that the instance profile associated with the instance has the necessary permissions to write to that bucket.

13. In the SNS notifications section, choose Enable SNS Notifications.
14. In the IAM role section, choose the SNS role to trigger Amazon SNS notifications.
15. In the SNS topic section, type or paste the Amazon SNS topic ARN to be used.
16. In the Event type section, choose the events for which you want to receive notifications.
17. In the Notification type section, choose to receive notifications for each copy of a command sent to multiple instances (invocations) or the command summary.
18. In the Input Parameters section, enter the required parameters based on the Command document you chose.
19. Choose Register Run Command task.
20. After the next execution of your maintenance window, check your email for a message from Amazon SNS and open the email. Amazon SNS can take a few minutes to send the mail.

Registering a Run Command task to a maintenance window that returns notifications (CLI)

Use the following procedure to register a Run Command task that is configured to return status notifications to your maintenance window using the AWS CLI.

To register a Run Command task with your maintenance window that returns notifications (CLI)

Note
To better manage your task options, this procedure uses the command option --cli-input-json, with option values stored in a JSON file.

1. On your local machine, create a file named RunCommandTask.json.
2. Paste the following contents into the file:

```json
{
   "Name": "Name",
   "Description": "Description",
   "WindowId": "mw-0c50858d01EXAMPLE",
   "ServiceRoleArn": "arn:aws:iam::111122223333:role/MaintenanceWindowIAMRole",
   "MaxConcurrency": "1",
   "MaxErrors": "1",
   "Priority": 3,
   "Targets": [
      {
         "Key": "WindowTargetIds",
         "Values": [
            "e32ee6c2-646c-4f4b-8ed1-205fbEXAMPLE"
         ]
      }
   ],
   "TaskType": "RUN_COMMAND",
   "TaskArn": "CommandDocumentName",
   "TaskInvocationParameters": {
      "RunCommand": {
         "Comment": "Comment",
         "TimeoutSeconds": 3600,
      }
   }
}
```

3. Replace the example values with information about your own resources.

You can also restore options we have omitted from this example if you want to use them. For example, you can save command output to an S3 bucket.

For more information, see register-task-with-maintenance-window.

4. Save the file.

5. In the directory on your local machine where you saved the file, run the following command:

```
aws ssm register-task-with-maintenance-window --cli-input-json file://RunCommandTask.json
```

**Important**

Be sure to include file:// before the file name. It is required in this command.

If successful, the command returns information similar to the following:

```
{
  "WindowTaskId": "j2l8d5b5c-mw66-tk4d-r3g9-1d4d1EXAMPLE"
}
```

6. After the next execution of your maintenance window, check your email for a message from Amazon SNS and open the email. Amazon SNS can take a few minutes to send the mail.

For more information about registering tasks for maintenance window from the command line, see Amazon EC2 Systems Manager API Reference and the Systems Manager AWS CLI Reference.
Taggable Systems Manager resources

A tag is a label that you assign to an AWS resource. Each tag consists of a key and a value, both of which you define.

Tags enable you to categorize your AWS resources in different ways, such as by purpose, owner, or environment. For example, if you want to organize and manage your resources according to whether they're used for development or production, you might tag some of them with the key Environment and the value Production. You can then perform various types of queries for resources tagged "Key=Environment,Values=Production". For example, you could define a set of tags for your account's managed instances that help you track or target instances by operating system and environment, such as your SUSE Linux Enterprise Server grouped as development, staging, and production. You can also perform operations on resources by specifying this key-value pair in your commands, such as running an update script on all instances in the group or reviewing the status of those instances.

You can use the tags applied to your Systems Manager in various operations. For example, you can target only managed instances that are tagged with a specified tag key-value pair when you run a command (p. 833) or assign targets to a maintenance window (p. 635). You can also restrict access to your resources (p. 1115) based on the tags applied to them.

Going further, you can create resource groups by specifying the same tags for AWS resources of various types, not only the same type. After that, you can use the Systems Manager Resource Groups capability to view information about which resources in a group are compliant and working correctly and which resources require action. The information you view pertains to all types of AWS resources that can be added to a resource group, not only supported Systems Manager resource types. For more information, see Resource Groups in AWS Systems Manager (p. 170) and Viewing operations data for AWS Resource Groups (p. 172).

The remainder of this chapter describes how to add and remove tags from Systems Manager resources.

Topics

- Taggable Systems Manager resources (p. 1163)
- Tagging Systems Manager documents (p. 1164)
- Tagging maintenance windows (p. 1168)
- Tagging managed instances (p. 1172)
- Tagging OpsItems (p. 1177)
- Tagging Systems Manager parameters (p. 1180)
- Tagging patch baselines (p. 1184)

Taggable Systems Manager resources

You can apply tags to the following Systems Manager resource types.

- Documents
- Maintenance windows
- Managed instances
- OpsItems
- Parameters
- Patch baselines
Each of these types except OpsItems can be added to a resource group.

Depending on the resource type, you can use tags to identify which resources should be included in an operation. For example, you can tag a group of managed instances and then run a maintenance window task that targets only instances with that key-value pair.

You can also restrict user access to these resource types by creating IAM policies that specify the tags that a user can access and attaching the policy to user accounts or groups. The following are a few examples of restricting resource access using tags.

- You can apply a tag to a set of custom Systems Manager documents and then create a user policy that grants access to documents with that tag but no others (or that prohibits access to only those documents).
- You can assign tags to OpsItems and then create IAM policies that limit which users or groups have access to view or update those resources. For example, organization directors could be granted full access to all OpsItems, but software developers and support engineers could be granted access only to the projects or client segments they are responsible for.
- You can apply a common tag to resources of all six supported types and create an IAM policy that grants access to only those resources, such as Key=Project,Value=ProjectA or Key=Environment,Value=Development. You can even grant access to only resources to which both tag pairs have been assigned. This lets you, for example, restrict users to working only with resources for ProjectA in the Development environment.

You can use the Systems Manager Resource Groups console, the console for the supported resource types (for example, the Maintenance Windows console or OpsCenter console), the AWS CLI, and the AWS Tools for PowerShell. You can add tags when you create or update a resource. For example, you can use the AWS CLI add-tags-to-resource command to add tags to any of the supported Systems Manager resource types after they have been created. You can use the remove-tags-from-resource command to remove them.

---

## Tagging Systems Manager documents

The topics in this section describe how to work with tags on Systems Manager documents.

**Topics**

- Creating documents with tags (p. 1164)
- Adding tags to existing documents (p. 1164)
- Removing tags from Systems Manager documents (p. 1166)

### Creating documents with tags

You can add tags to custom Systems Manager documents at the time you create them.

For information, see the following topics:

- Create an SSM document (console) (p. 1086)
- Create an SSM document (command line) (p. 1087)

### Adding tags to existing documents

You can add tags to custom Systems Manager documents that you own by using the Systems Manager console or the command line.
Adding tags to existing documents

Adding tags to an existing SSM document (console)

2. In the navigation pane, choose Documents.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Documents in the navigation pane.
3. Choose the Owned by me tab.
4. Choose the name of the document to add tags to, and then choose the Details tab.
5. In the Tags section, choose Edit, and then add one or more key-value tag pairs.
6. Choose Save.

Adding tags to an existing SSM document (command line)

To add tags to an existing document (command line)

1. Using your preferred command line tool, run the following command to view the list of documents that you can tag.

   Linux
   ```bash
   aws ssm list-documents
   ```

   Windows
   ```bash
   aws ssm list-documents
   ```

   PowerShell
   ```bash
   Get-SSMDocumentList
   ```

   Note the name of a document that you want to tag.
2. Run the following command to tag a document.

   Linux
   ```bash
   aws ssm add-tags-to-resource
   --resource-type "Document"
   --resource-id "document-name"
   --tags "Key=tag-key,Value=tag-value"
   ```

   Windows
   ```bash
   aws ssm add-tags-to-resource
   --resource-type "Document"
   ```
Removing tags from Systems Manager documents

You can use the Systems Manager console or the command line to remove tags from Systems Manager documents.

Topics

- Removing tags from Systems Manager documents (console) (p. 1167)
Removing tags from Systems Manager documents (command line) (p. 1167)

Removing tags from Systems Manager documents (console)

2. In the navigation pane, choose Documents.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Documents in the navigation pane.
3. Choose the Owned by me tab.
4. Choose the name of the document to remove tags from, and then choose the Details tab.
5. In the Tags section, choose Edit, and then choose Remove next to the tag pair you no longer need.
6. Choose Save.

Removing tags from Systems Manager documents (command line)

1. Using your preferred command line tool, run the following command to list the documents in your account.

   **Linux**
   
   ```bash
   aws ssm list-documents
   ```

   **Windows**
   
   ```bash
   aws ssm list-documents
   ```

   **PowerShell**
   
   ```bash
   Get-SSMDocumentList
   ```

   Note the name of a document from which you want to remove tags.

2. Run the following command to remove tags from a document.

   **Linux**
   
   ```bash
   aws ssm remove-tags-from-resource \
   --resource-type "Document" \
   --resource-id "document-name" \
   --tag-key "tag-key"
   ```

   **Windows**
   
   ```bash
   aws ssm remove-tags-from-resource ^
   --resource-type "Document" ^
   --resource-id "document-name" ^
   --tag-key "tag-key"
   ```
Tagging maintenance windows

The topics in this section describe how to work with tags on maintenance windows.

Topics
- Creating maintenance windows with tags (p. 1168)
- Adding tags to existing maintenance windows (p. 1169)
- Removing tags from maintenance windows (p. 1171)

Creating maintenance windows with tags

You can add tags to maintenance windows at the time you create them.

For information, see the following topics:
- Create a maintenance window (console) (p. 634)
- ??? (p. 642)Create and configure a maintenance window (AWS CLI)
Adding tags to existing maintenance windows

You can add tags to maintenance windows that you own by using the Systems Manager console or the command line.

**Topics**

- Adding tags to an existing maintenance window (console) (p. 1169)
- Adding tags to an existing maintenance window (AWS CLI) (p. 1169)
- Tag a maintenance window (AWS Tools for PowerShell) (p. 1170)

**Adding tags to an existing maintenance window (console)**

2. In the navigation pane, choose **Maintenance Windows**.
   - or -
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Maintenance Windows**.
3. Choose the name of a maintenance window you have already created, and then choose the **Tags** tabs.
4. Choose **Edit tags**, and then choose **Add tag**.
5. For **Key**, enter a key for the tag, such as **Environment**.
6. For **Value**, enter a value for the tag, such as **Test**.
7. Choose **Save changes**.

**Adding tags to an existing maintenance window (AWS CLI)**

1. Using your preferred command line tool, run the following command to view the list of maintenance windows that you can tag.

   ```bash
   aws ssm describe-maintenance-windows
   ```

   Note the ID of a maintenance window that you want to tag.
2. Run the following command to tag a maintenance window.

   **Linux**

   ```bash
   aws ssm add-tags-to-resource
   --resource-type "MaintenanceWindow"
   --resource-id "window-id"
   --tags "Key=tag-key,Value=tag-value"
   ```

   **Windows**

   ```bash
   aws ssm add-tags-to-resource
   --resource-type "MaintenanceWindow"
   --resource-id "window-id"
   --tags "Key=tag-key,Value=tag-value"
   ```
Adding tags to existing maintenance windows

If successful, the command has no output.

*windows-id* is the ID of the maintenance window you want to tag, such as mw-0c50858d01EXAMPLE.

*tag-key* is the name of a custom key you supply. For example, Environment or Project.

*tag-value* is the custom content for the value you want to supply for that key. For example, Production or Q321.

3. Run the following command to verify the maintenance window tags.

    **Linux**

    ```bash
    aws ssm list-tags-for-resource --resource-type "MaintenanceWindow" --resource-id "windows-id"
    ```

    **Windows**

    ```bash
    aws ssm list-tags-for-resource ^
    --resource-type "MaintenanceWindow" ^
    --resource-id "windows-id"
    ```

Tag a maintenance window (AWS Tools for PowerShell)

1. Run the following command to list maintenance windows that you can tag.

   ```powershell
   Get-SSMMaintenanceWindow
   ```

2. Run the following commands to tag a maintenance window.

   ```powershell
   $tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
   $tag.Key = "tag-key"
   $tag.Value = "tag-value"
   Add-SSMResourceTag -ResourceType "MaintenanceWindow" -ResourceId "window-id" -Tag $tag
   ```

   *window-id* the ID of the maintenance window you want to tag.

   *tag-key* is the name of a custom key you supply. For example, Environment or Project.

   *tag-value* is the custom content for the value you want to supply for that key. For example, Production or Q321.

3. Run the following command to verify the maintenance window tags.

   ```powershell
   Get-SSMResourceTag `
Removing tags from maintenance windows

You can use the Systems Manager console or the command line to remove tags from maintenance windows.

Topics

- Removing tags from maintenance windows (console) (p. 1171)
- Removing tags from maintenance windows (command line) (p. 1171)

Removing tags from maintenance windows (console)

2. In the navigation pane, choose Maintenance Windows.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon to open the navigation pane, and then choose Maintenance Windows.
3. Choose the name of the maintenance window to remove tags from, and then choose Tags tab.
4. Choose Edit tags, and then choose Remove tag next to the tag pair you no longer need.
5. Choose Save changes.

Removing tags from maintenance windows (command line)

1. Using your preferred command line tool, run the following command to list the maintenance windows in your account.
   Linux
   ```bash
   aws ssm describe-maintenance-windows
   ```
   Windows
   ```bash
   aws ssm describe-maintenance-windows
   ```
   PowerShell
   ```bash
   Get-SSMMaintenanceWindows
   ```
   Note the ID of a maintenance window from which you want to remove tags.
2. Run the following command to remove tags from a maintenance window.
   Linux
   ```bash
   aws ssm remove-tags-from-resource
   --resource-type "MaintenanceWindow"
   --resource-id "window-id"
   ```
Tagging managed instances

The topics in this section describe how to work with tags on managed instances.

A managed instance is any machine configured for AWS Systems Manager. This includes EC2 instances, as well as on-premises servers or virtual machines (VMs) that you have configured to manage using Systems Manager in a hybrid environment.

The instructions in this topic are applicable to any machine that is being managed using Systems Manager.
Creating or activating managed instances with tags

You can add tags to EC2 instances at the time you create them. You can add tags to on-premises servers and virtual machines (VMs) at the time you activate them.

For information, see the following topics:

- For EC2 instances, see Tagging your Amazon EC2 resources in the Amazon EC2 User Guide for Linux Instances. (Content applies to both EC2 instances for Linux and for Windows)
- For on-premises servers and VMs, see Create a managed-instance activation for a hybrid environment (p. 48).

Adding tags to existing managed instances

You can add tags to managed instances by using the Systems Manager console or the command line.

Topics

- Adding tags to an existing managed instance (console) (p. 1173)
- Adding tags to an existing managed instance (command line) (p. 1173)

Adding tags to an existing managed instance (console)

2. In the navigation pane, choose Managed Instances.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Managed Instances.
3. Choose the name of the managed instance to add tags to, and then choose the Tags tab.
4. In the Tags section, choose Edit, and then add one or more key-value tag pairs.
5. Choose Save.

Adding tags to an existing managed instance (command line)

To add tags to an existing managed instance (command line)

1. Using your preferred command line tool, run the following command to view the list of managed instances that you can tag.
   
   ```
   Linux
   aws ssm describe-instance-information
   ```
Adding tags to existing managed instances

Windows

```bash
aws ssm describe-instance-information
```

PowerShell

```powershell
Get-SSMInstanceInformation
```

Note the ID of a managed instance that you want to tag.

**Note**

Machines that have been registered for use with Systems Manager in a hybrid environment begin with `mi-`, such as `mi-0471e04240EXAMPLE`; EC2 instances have IDs that begin with `i-`, such as `i-02573cafcfEXAMPLE`.

2. Run the following command to tag a managed instance.

**Linux**

```bash
aws ssm add-tags-to-resource \
  --resource-type "ManagedInstance" \
  --resource-id "instance-id" \
  --tags "Key=tag-key,Value=tag-value"
```

**Windows**

```bash
aws ssm add-tags-to-resource ^ \
  --resource-type "ManagedInstance" ^ \
  --resource-id "instance-id" ^ \
  --tags "Key=tag-key,Value=tag-value"
```

**PowerShell**

```powershell
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
$tag.Key = "tag-key"
$tag.Value = "tag-value"
Add-SSMResourceTag \
  -ResourceType "ManagedInstance" \
  -ResourceId "instance-id" \
  -Tag $tag \
  -Force
```

*tag-key* is the name of a custom key you supply. For example, *Region* or *Quarter*.

*tag-value* is the custom content for the value you want to supply for that key. For example, *West* or *Q321*.

*instance-id* is the ID of the managed instance you want to tag.

If successful, the command has no output.
Removing tags from managed instances (console)

2. In the navigation pane, choose Managed Instances.
3. Choose the name of the managed instance to remove tags from, and then choose the Tags tab.
4. In the Tags section, choose Edit, and then choose Remove next to the tag pair you no longer need.
5. Choose Save.

Removing tags from managed instances (command line)

1. Using your preferred command line tool, run the following command to list the managed instances in your account.

Linux

```
aws ssm describe-instance-information
```
Removing tags from managed instances

Windows

aws ssm describe-instance-information

PowerShell

Get-SSMInstanceInformation

Note the name of a managed instance from which you want to remove tags.

2. Run the following command to remove tags from a managed instance.

Linux

aws ssm remove-tags-from-resource \
   --resource-type "ManagedInstance" \
   --resource-id "instance-id" \
   --tag-key "tag-key"

Windows

aws ssm remove-tags-from-resource ^
   --resource-type "ManagedInstance" ^
   --resource-id "instance-id" ^
   --tag-key "tag-key"

PowerShell

Remove-SSMResourceTag `\n   -ResourceId "instance-id" `\n   -ResourceType "ManagedInstance" `\n   -TagKey "tag-key" `\n   -Force

instance-id is the name of the managed instance from which you want to remove tags.

tag-key is the name of a key assigned to the managed instance. For example, Environment or Quarter.

If successful, the command has no output.

3. Run the following command to verify the managed instance tags.

Linux

aws ssm list-tags-for-resource \
   --resource-type "ManagedInstance" \
   --resource-id "instance-id"

Windows

aws ssm list-tags-for-resource ^
   --resource-type "ManagedInstance" ^
   --resource-id "instance-id"
Tagging OpsItems

The topics in this section describe how to work with tags on OpsItems.

Topics
- Creating OpsItems with tags (p. 1177)
- Adding tags to existing OpsItems (p. 1177)
- Removing tags from Systems Manager OpsItems (p. 1179)

Creating OpsItems with tags

You can add tags to custom Systems Manager OpsItems at the time you create them if you use a command line tool.

For information, see the following topic:
- To manually create an OpsItem (AWS CLI) (p. 151)

Adding tags to existing OpsItems

You can add tags to OpsItems by using a command line tool.

Topics
- Adding tags to an existing OpsItem (command line) (p. 1177)

Adding tags to an existing OpsItem (command line)

To add tags to an existing OpsItem (command line)

1. Using your preferred command line tool, run the following command to view the list of OpsItem that you can tag.

   Linux
   ```
   aws ssm describe-ops-items
   ```

   Windows
   ```
   aws ssm describe-ops-items
   ```

   PowerShell
   ```
   Get-SSMOpsItemSummary
   ```
Note the ID of an OpsItem that you want to tag.

2. Run the following command to tag an OpsItem.

Linux

```bash
aws ssm add-tags-to-resource \
   --resource-type "OpsItem" \
   --resource-id "ops-item-id" \
   --tags "Key=tag-key,Value=tag-value"
```

Windows

```bash
aws ssm add-tags-to-resource ^
   --resource-type "OpsItem" ^
   --resource-id "ops-item-id" ^
   --tags "Key=tag-key,Value=tag-value"
```

PowerShell

```powershell
$tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag

$tag.Key = "tag-key"

$tag.Value = "tag-value"

Add-SSMResourceTag `n-ResourceType "OpsItem" `n-ResourceId "ops-item-id" `n-Tag $tag `n-Force
```

tag-key is the name of a custom key you supply. For example, Region or Quarter.
tag-value is the custom content for the value you want to supply for that key. For example, West or Q321.

ops-item-id is the ID of the OpsItem you want to tag.

If successful, the command has no output.

3. Run the following command to verify the OpsItem tags.

Linux

```bash
aws ssm list-tags-for-resource \
   --resource-type "OpsItem" \
   --resource-id "ops-item-id"
```

Windows

```bash
aws ssm list-tags-for-resource ^
   --resource-type "OpsItem" ^
```
Removing tags from Systems Manager OpsItems

You can use a command line tool to remove tags from Systems Manager OpsItems.

Topics
- Removing tags from OpsItems (command line) (p. 1179)

Removing tags from OpsItems (command line)

1. Using your preferred command line tool, run the following command to list the OpsItems in your account.

   Linux
   ```bash
   aws ssm describe-ops-items
   ```

   Windows
   ```bash
   aws ssm describe-ops-items
   ```

   PowerShell
   ```bash
   Get-SSMOPSItemSummary
   ```

   Note the name of an OpsItem from which you want to remove tags.

2. Run the following command to remove tags from an OpsItem.

   Linux
   ```bash
   aws ssm remove-tags-from-resource \    --resource-type "OpsItem" \    --resource-id "ops-item-id" \    --tag-key "tag-key"
   ```

   Windows
   ```bash
   aws ssm remove-tags-from-resource ^    --resource-type "OpsItem" ^    --resource-id "ops-item-id" ^    --tag-key "tag-key"
   ```
Tagging Systems Manager parameters

The topics in this section describe how to work with tags on Systems Manager parameters.

Topics
- Creating parameters with tags (p. 1180)
- Adding tags to existing parameters (p. 1181)
- Removing tags from Systems Manager parameters (p. 1182)

Creating parameters with tags

You can add tags to Systems Manager parameters at the time you create them.

For information, see the following topics:
- Create a Systems Manager parameter (console) (p. 249)
- Create a Systems Manager parameter (AWS CLI) (p. 251)
- Create a Systems Manager parameter (Tools for Windows PowerShell) (p. 255)
Adding tags to existing parameters

You can add tags to custom Systems Manager parameters that you own by using the Systems Manager console or the command line.

**Topics**
- Adding tags to an existing parameter (console) (p. 1181)
- Adding tags to an existing parameter (AWS CLI) (p. 1181)
- Adding tags to an existing parameter (AWS Tools for PowerShell) (p. 1182)

Adding tags to an existing parameter (console)

2. In the navigation pane, choose **Parameter Store**.
   - or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Parameter Store**.
3. Choose the name of a parameter you have already created, and then choose the **Tags** tab.
4. In the first box, enter a key for the tag, such as **Environment**.
5. In the second box, enter a value for the tag, such as **Test**.
6. Choose **Save**.

Adding tags to an existing parameter (AWS CLI)

1. Using your preferred command line tool, run the following command to view the list of parameters that you can tag.
   ```bash
   aws ssm describe-parameters
   ```
   Note the name of a parameter that you want to tag.
2. Run the following command to tag a parameter.
   ```bash
   aws ssm add-tags-to-resource --resource-type "Parameter" --resource-id "parameter-name" --tags "Key=tag-key,Value=tag-value"
   ```
   If successful, the command has no output.
   - **parameter-name** is the name of the SSM parameter you want to tag.
   - **tag-key** is the name of a custom key you supply. For example, **Environment** or **Project**.
   - **tag-value** is the custom content for the value you want to supply for that key. For example, **Production** or **Q321**.
3. Run the following command to verify the parameter tags.
   ```bash
   aws ssm list-tags-for-resource --resource-type "Parameter" --resource-id "parameter-name"
   ```
Adding tags to an existing parameter (AWS Tools for PowerShell)

1. Run the following command to list parameters that you can tag.

   ```powershell
   Get-SSMParameterList
   ```

2. Run the following commands to tag a parameter.

   ```powershell
   $tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
   $tag.Key = "tag-key"
   $tag.Value = "tag-value"
   Add-SSMResourceTag -ResourceType "Parameter" -ResourceId "parameter-name" -Tag $tag -Force
   ```

   * parameter-name the name of the SSM parameter you want to tag.
   * tag-key is the name of a custom key you supply. For example, Environment or Project.
   * tag-value is the custom content for the value you want to supply for that key. For example, Production or Q321.

3. Run the following command to verify the parameter tags.

   ```powershell
   Get-SSMResourceTag -ResourceType "Parameter" -ResourceId "parameter-name"
   ```

Removing tags from Systems Manager parameters

You can use the Systems Manager console or the command line to remove tags from Systems Manager parameters.

**Topics**

- Removing tags from Systems Manager parameters (console) (p. 1182)
- Removing tags from Systems Manager parameters (command line) (p. 1183)

**Removing tags from Systems Manager parameters (console)**

2. In the navigation pane, choose Parameter Store.
   -or-
Removing tags from Systems Manager parameters

If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose **Parameter Store**.

3. Choose the name of the parameter to remove tags from, and then choose the **Tags** tab.
4. Choose **Remove** next to the tag pair you no longer need.
5. Choose **Save**.

**Removing tags from Systems Manager parameters (command line)**

1. Using your preferred command line tool, run the following command to list the parameters in your account.
   
   **Linux**
   ```bash
   aws ssm describe-parameters
   ```
   
   **Windows**
   ```bash
   aws ssm describe-parameters
   ```
   
   **PowerShell**
   ```bash
   Get-SSMParameterList
   ```

   Note the name of a parameter from which you want to remove tags.

2. Run the following command to remove tags from a parameter.
   
   **Linux**
   ```bash
   aws ssm remove-tags-from-resource \
   --resource-type "Parameter" \
   --resource-id "parameter-name" \ 
   --tagkey "tag-key"
   ```
   
   **Windows**
   ```bash
   aws ssm remove-tags-from-resource ^
   --resource-type "Parameter" ^
   --resource-id "parameter-name" ^
   --tag-key "tag-key"
   ```
   
   **PowerShell**
   ```bash
   Remove-SSMResourceTag
   -ResourceId "parameter-name"
   -ResourceType "Parameter"
   -TagKey "tag-key"
   ```

   *parameter-name* is the name of the SSM parameter from which you want to remove a tag.
tag-key is the name of a key assigned to the parameter. For example, Environment or Quarter.

If successful, the command has no output.

3. Run the following command to verify the document tags.

   **Linux**
   ```bash
   aws ssm list-tags-for-resource \
   --resource-type "Parameter" \
   --resource-id "parameter-name"
   ```

   **Windows**
   ```bash
   aws ssm list-tags-for-resource ^
   --resource-type "Parameter" ^
   --resource-id "parameter-name"
   ```

   **PowerShell**
   ```powershell
   Get-SSMResourceTag ^
   -ResourceType "Parameter" ^
   -ResourceId "parameter-name"
   ```

---

**Tagging patch baselines**

The topics in this section describe how to work with tags on patch baselines.

**Topics**

- Creating patch baselines with tags (p. 1184)
- Adding tags to existing patch baselines (p. 1184)
- Removing tags from patch baselines (p. 1186)

---

**Creating patch baselines with tags**

You can add tags to Systems Manager parameters at the time you create them.

For information, see the following topics:

- Create a custom patch baseline (p. 959)
- Create a patch baseline (p. 974)
- Create a patch baseline with custom repositories for different OS versions (p. 975)

---

**Adding tags to existing patch baselines**

You can add tags to patch baselines that you own by using the Systems Manager console or the command line.

**Topics**

- Adding tags to an existing patch baseline (console) (p. 1185)
• Adding tags to an existing parameter (AWS CLI) (p. 1185)
• Tag a patch baseline (AWS Tools for PowerShell) (p. 1186)

Adding tags to an existing patch baseline (console)
2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Patch Manager.
3. Choose the name of a custom patch baseline you have already created, scroll down to the Tags table section, and then choose Edit tags.
4. Choose Add tags.
5. For Key, enter a key for the tag, such as Environment.
6. For Value, enter a value for the tag, such as Test.
7. Choose Save changes.

Adding tags to an existing parameter (AWS CLI)
1. Using your preferred command line tool, run the following command to view the list of parameters that you can tag.

   ```bash
   aws ssm describe-patch-baselines
   ```

   Note the ID of a patch baseline that you want to tag.
2. Run the following command to tag a patch baseline.

   **Linux**

   ```bash
   aws ssm add-tags-to-resource \
   --resource-type "PatchBaseline" \ 
   --resource-id "baseline-id" \ 
   --tags "Key=tag-key,Value=tag-value"
   ```

   **Windows**

   ```bash
   aws ssm add-tags-to-resource ^
   --resource-type "PatchBaseline" ^
   --resource-id "baseline-id" ^
   --tags "Key=tag-key,Value=tag-value"
   ```

   If successful, the command has no output.

   *baseline-id* is the ID of the patch baseline you want to tag, such as pb-0c10e65780EXAMPLE.

   *tag-key* is the name of a custom key you supply. For example, Environment or Project.

   *tag-value* is the custom content for the value you want to supply for that key. For example, Production or Q321.
3. Run the following command to verify the patch baseline tags.
Tag a patch baseline (AWS Tools for PowerShell)

1. Run the following command to list patch baseline that you can tag.

   ```powershell
   Get-SSMPatchBaseline
   ```

2. Run the following commands to tag a patch baseline.

   ```powershell
   $tag = New-Object Amazon.SimpleSystemsManagement.Model.Tag
   $tag.Key = "tag-key"
   $tag.Value = "tag-value"
   Add-SSMResourceTag
   -ResourceType "PatchBaseline"
   -ResourceId "baseline-id"
   -Tag $tag
   -Force
   ```

   - `patch-baseline-name` the name of the SSM parameter you want to tag.
   - `tag-key` is the name of a custom key you supply. For example, `Environment` or `Project`.
   - `tag-value` is the custom content for the value you want to supply for that key. For example, `Production` or `Q321`.

3. Run the following command to verify the patch baseline tags.

   ```powershell
   Get-SSMResourceTag
   -ResourceType "PatchBaseline"
   -ResourceId "baseline-id"
   ```

Removing tags from patch baselines

You can use the Systems Manager console or the command line to remove tags from patch baseline.

**Topics**

- **Removing tags from patch baseline (console)** (p. 1187)
Removing tags from patch baselines (command line) (p. 1187)

Removing tags from patch baseline (console)
2. In the navigation pane, choose Patch Manager.
   -or-
   If the AWS Systems Manager home page opens first, choose the menu icon (≡) to open the navigation pane, and then choose Patch Manager.
3. Choose the name of the patch baseline to remove tags from, scroll down to the Tags table section, and then choose Edit tags tab.
4. Choose Remove tag next to the tag pair you no longer need.
5. Choose Save changes.

Removing tags from patch baselines (command line)
1. Using your preferred command line tool, run the following command to list the patch baselines in your account.
   - Linux
   ```bash
   aws ssm describe-patch-baselines
   ```
   - Windows
   ```cmd
   aws ssm describe-patch-baselines
   ```
   - PowerShell
   ```powershell
   Get-SSMPatchBaseline
   ```
   Note the ID of a patch baseline from which you want to remove tags.
2. Run the following command to remove tags from a patch baseline.
   - Linux
   ```bash
   aws ssm remove-tags-from-resource
   --resource-type "PatchBaseline"
   --resource-id "baseline-id"
   --tag-key "tag-key"
   ```
   - Windows
   ```cmd
   aws ssm remove-tags-from-resource
   --resource-type "PatchBaseline"
   --resource-id "baseline-id"
   --tag-key "tag-key"
   ```
**PowerShell**

```powershell
Remove-SSMResourceTag
  -ResourceType "PatchBaseline"
  -ResourceId "baseline-id"
  -TagKey "tag-key"
```

*baseline-id* is the ID of the patch baseline you want to tag, such as pb-0c10e65780EXAMPLE.

*tag-key* is the name of a key assigned to the patch baseline. For example, *Environment* or *Quarter*.

If successful, the command has no output.

3. Run the following command to verify the patch baseline tags.

**Linux**

```bash
aws ssm list-tags-for-resource
  --resource-type "PatchBaseline"
  --resource-id "baseline-id"
```

**Windows**

```bash
aws ssm list-tags-for-resource ^
  --resource-type "PatchBaseline" ^
  --resource-id "baseline-id"
```

**PowerShell**

```powershell
Get-SSMResourceTag
  -ResourceType "PatchBaseline"
  -ResourceId "baseline-id"
```
Reference: Cron and rate expressions for Systems Manager

When you create an AWS Systems Manager maintenance window or a State Manager association, you specify a schedule for when the window or the association should run. You can specify a schedule in the form of either a time-based entry, called a **cron expression**, or a frequency-based entry, called a **rate expression**. For maintenance windows, you can also specify a time stamp in Coordinated Universal Time (UTC) format when you create a maintenance window so that it runs once at the specified time.
When you create either type of resource programmatically or by using a command line tool such as the AWS CLI, you must specify a schedule parameter with a valid cron or rate expression (or time stamp for maintenance windows) in the correct format.

When you use the AWS Systems Manager console to create a maintenance window or association, you can specify a schedule using a valid cron or rate expression. You can also use tools in the user interface that simplify the process of creating your schedule.

Maintenance Window Examples

To create maintenance windows using the AWS CLI, you include the `--schedule` parameter with a cron or rate expression or a time stamp. For example:

```bash
aws ssm create-maintenance-window --name "My-Cron-Maintenance-Window" --schedule "cron(0 16 ? * TUE *)" --schedule-timezone "America/Los_Angeles" --start-date 2019-01-01T00:00:00-08:00 --end-date 2019-06-30T00:00:00-08:00 --duration 4 --cutoff 1
```

```bash
aws ssm create-maintenance-window --name "My-Rate-Maintenance-Window" --schedule "rate(7 days)" --duration 4 --schedule-timezone "America/Los_Angeles" --cutoff 1
```

```bash
aws ssm create-maintenance-window --name "My-TimeStamp-Maintenance-Window" --schedule "at(2019-07-07T13:15:30)" --duration 4 --schedule-timezone "America/Los_Angeles" --cutoff 1
```

Association Examples

To create State Manager associations using the AWS CLI, you include the `--schedule-expression` parameter with a cron or rate expression. For example:

```bash
aws ssm create-association --association-name "My-Cron-Association" --schedule-expression "cron(0 0 2 ? * SUN *)" --targets Key=tag:ServerRole,Values=WebServer --name AWS-UpdateSSMAgent
```

```bash
aws ssm create-association --association-name "My-Rate-Association" --schedule-expression "rate(7 days)" --targets Key=tag:ServerRole,Values=WebServer --name AWS-UpdateSSMAgent
```

Topics

- General information about cron and rate expressions (p. 1190)
- Cron and rate expressions for associations (p. 1193)
- Cron and rate expressions for maintenance windows (p. 1194)

General information about cron and rate expressions

Cron expressions for Systems Manager have six required fields. A seventh field, the `Seconds` field (the first in a cron expression), is optional. Fields are separated by a space.

Cron expression examples

<table>
<thead>
<tr>
<th>Minutes</th>
<th>Hours</th>
<th>Day of month</th>
<th>Month</th>
<th>Day of week</th>
<th>Year</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 10:00 am (UTC) every day</td>
</tr>
</tbody>
</table>
### General information about cron and rate expressions

<table>
<thead>
<tr>
<th>Minutes</th>
<th>Hours</th>
<th>Day of month</th>
<th>Month</th>
<th>Day of week</th>
<th>Year</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 12:15 PM (UTC) every day</td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run at 6:00 PM (UTC) every Monday through Friday</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>1</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run at 8:00 AM (UTC) every 1st day of the month</td>
</tr>
<tr>
<td>0/15</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>Run every 15 minutes</td>
</tr>
<tr>
<td>0/10</td>
<td>*</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 10 minutes Monday through Friday</td>
</tr>
<tr>
<td>0/5</td>
<td>8-17</td>
<td>?</td>
<td>*</td>
<td>MON-FRI</td>
<td>*</td>
<td>Run every 5 minutes Monday through Friday between 8:00 AM and 5:55 PM (UTC)</td>
</tr>
</tbody>
</table>

**Supported Values**

The following table shows supported values for required cron entries.

**Note**

Cron expressions for associations do not support all these values. For information, see Cron and rate expressions for associations (p. 1193).

**Supported values for cron expressions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Wildcards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minutes</td>
<td>0-59</td>
<td>, - * /</td>
</tr>
<tr>
<td>Hours</td>
<td>0-23</td>
<td>, - * /</td>
</tr>
<tr>
<td>Day-of-month</td>
<td>1-31</td>
<td>, - * ? / L W</td>
</tr>
</tbody>
</table>
### Wildcards

The following table shows the wildcard values that cron expressions support.

#### Supported wildcards for cron expressions

<table>
<thead>
<tr>
<th>Wildcard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>,</td>
<td>The , (comma) wildcard includes additional values. In the Month field, JAN,FEB,MAR would include January, February, and March.</td>
</tr>
<tr>
<td>-</td>
<td>The - (dash) wildcard specifies ranges. In the Day field, 1-15 would include days 1 through 15 of the specified month.</td>
</tr>
<tr>
<td>*</td>
<td>The * (asterisk) wildcard includes all values in the field. In the Hours field, * would include every hour.</td>
</tr>
<tr>
<td>/</td>
<td>The / (forward slash) wildcard specifies increments. In the Minutes field, you could enter 1/10 to specify every tenth minute, starting from the first minute of the hour. So 1/10 specifies the first, 11th, 21st, and 31st minute, and so on.</td>
</tr>
<tr>
<td>?</td>
<td>The ? (question mark) wildcard specifies one or another. In the Day-of-month field you could enter 7 and if you didn’t care what day of the week the 7th was, you could enter ? in the Day-of-week field.</td>
</tr>
<tr>
<td>L</td>
<td>The L wildcard in the Day-of-month or Day-of-week fields specifies the last day of the month or week.</td>
</tr>
<tr>
<td>W</td>
<td>The W wildcard in the Day-of-month field specifies a weekday. In the Day-of-month field, 3W specifies the day closest to the third weekday of the month.</td>
</tr>
</tbody>
</table>

### Note

Cron expressions that lead to rates faster than five (5) minute are not supported. Support for specifying both a day-of-week and a day-of-month value is not complete. You must currently use the question mark (?) character in one of these fields.

For more information about cron expressions, see [CRON expression](https://en.wikipedia.org/wiki/Cron) at the [Wikipedia website](https://en.wikipedia.org).
Rate Expressions

Rate expressions have the following two required fields. Fields are separated by white space.

**Required fields for rate expressions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>positive number, such as 1 or 15</td>
</tr>
<tr>
<td>Unit</td>
<td>minute</td>
</tr>
<tr>
<td></td>
<td>minutes</td>
</tr>
<tr>
<td></td>
<td>hour</td>
</tr>
<tr>
<td></td>
<td>hours</td>
</tr>
<tr>
<td></td>
<td>day</td>
</tr>
<tr>
<td></td>
<td>days</td>
</tr>
</tbody>
</table>

If the value is equal to 1, then the unit must be singular. Similarly, for values greater than 1, the unit must be plural. For example, rate(1 hour) and rate(5 hour) are not valid, but rate(1 hour) and rate(5 hours) are valid.

Cron and rate expressions for associations

This section includes examples of cron and rate expressions for State Manager associations. Before you create one of these expressions, be aware of the following restrictions.

- Associations only support the following cron expressions: every 1/2, 1, 2, 4, 8, or 12 hours; every day or every week at a specific time.
- Associations only support the following rate expressions: intervals of 30 minutes or greater and less than 31 days.
- If you specify the optional Seconds field, its value can only be 0 (zero). For example: `cron(0 */30 * * * ? *)`

**Note**

For an association that collects metadata for Systems Manager Inventory, we recommend using a rate expression.

The following table presents cron examples for associations using the required six fields.

**Cron examples for associations**

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>cron(0/30 * * * ? *)</td>
<td>Every 30 minutes</td>
</tr>
<tr>
<td>cron(0 0/1 * * ? *)</td>
<td>Every hour</td>
</tr>
<tr>
<td>cron(0 0/2 * * ? *)</td>
<td>Every 2 hours</td>
</tr>
<tr>
<td>cron(0 0/4 * * ? *)</td>
<td>Every 4 hours</td>
</tr>
<tr>
<td>cron(0 0/8 * * ? *)</td>
<td>Every 8 hours</td>
</tr>
</tbody>
</table>
Cron and rate expressions for maintenance windows

This section includes examples of cron and rate expressions for maintenance windows.

Unlike State Manager associations, maintenance windows support all cron and rate expressions. This includes support for values in the seconds field. (Note: Zero (0) is not supported in the seconds field.)

For example, the following 6-field cron expression runs a maintenance window at 9:30 AM every day:

```
cron(30 09 ? * * *)
```

By adding a value to the Seconds field, the following 7-field cron expression runs a maintenance window at 9:30:24 AM every day:

```
cron(24 30 09 ? * * *)
```

The following table provides additional 6-field cron examples for maintenance windows.

### Cron examples for maintenance windows

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>cron(0 2 ? * THU#3 *)</td>
<td>02:00 AM the third Thursday of every month</td>
</tr>
<tr>
<td>cron(15 10 ? * * *)</td>
<td>10:15 AM every day</td>
</tr>
<tr>
<td>cron(15 10 ? * MON-FRI *)</td>
<td>10:15 AM every Monday, Tuesday, Wednesday, Thursday and Friday</td>
</tr>
<tr>
<td>cron(0 2 L * ? *)</td>
<td>02:00 AM on the last day of every month</td>
</tr>
<tr>
<td>cron(15 10 ? * 6L *)</td>
<td>10:15 AM on the last Friday of every month</td>
</tr>
</tbody>
</table>

Here are some rate examples for associations.

### Rate examples for associations

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate(30 minutes)</td>
<td>Every 30 minutes</td>
</tr>
<tr>
<td>rate(1 hour)</td>
<td>Every hour</td>
</tr>
<tr>
<td>rate(5 hours)</td>
<td>Every 5 hours</td>
</tr>
<tr>
<td>rate(15 days)</td>
<td>Every 15 days</td>
</tr>
</tbody>
</table>

Cron and rate expressions for maintenance windows
The following table provides rate examples for maintenance windows.

### Rate examples for maintenance windows

<table>
<thead>
<tr>
<th>Example</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate(30 minutes)</td>
<td>Every 30 minutes</td>
</tr>
<tr>
<td>rate(1 hour)</td>
<td>Every hour</td>
</tr>
<tr>
<td>rate(5 hours)</td>
<td>Every 5 hours</td>
</tr>
<tr>
<td>rate(25 days)</td>
<td>Every 25 days</td>
</tr>
</tbody>
</table>

Reference: ec2messages, ssmmessages, and other API calls

If you monitor API calls, you might see calls to the following APIs.

- ec2messages:AcknowledgeMessage
- ec2messages:DeleteMessage
- ec2messages:FailMessage
- ec2messages:GetEndpoint
- ec2messages:GetMessages
- ec2messages:SendReply
- ssmmessages:CreateControlChannel
- ssmmessages:CreateDataChannel
- ssmmessages:OpenControlChannel
- ssmmessages:OpenDataChannel
- ssm:UpdateInstanceInformation
- ssm:ListInstanceAssociations
- ssm:DescribeInstanceProperties
- ssm:DescribeDocumentParameters

These special calls are used by Systems Manager for various operations.

**ec2messages API calls**

Calls to ec2messages:* APIs are calls to the Amazon Message Delivery Service endpoint. Systems Manager uses this endpoint to make calls from SSM Agent to the Systems Manager service in the cloud. This endpoint is required to send and receive commands. For more information, see Actions, Resources, and Condition Keys for Amazon Message Delivery Service.

**ssmmessages API calls**

Systems Manager uses the ssmmessages endpoint to make calls from SSM Agent to the Session Manager service in the cloud. This endpoint is required to create and delete session channels with the Session Manager service in the cloud.

**Instance-related API calls**
UpdateInstanceInformation: SSM Agent calls the Systems Manager service in the cloud every five minutes to provide heartbeat information. This call is necessary to maintain a heartbeat with the agent so that the service knows the agent is functioning as expected.

ListInstanceAssociations: The agent calls this API to see if a new Systems Manager State Manager association is available. This API is required for State Manager to function.

DescribeInstanceProperties and DescribeDocumentParameters: Systems Manager calls these APIs to render specific nodes in the Amazon EC2 console. The DescribeInstanceProperties API displays the Managed Instances node in the left navigation. The DescribeDocumentParameters API displays the Documents node in the left navigation.

Reference: Creating formatted date and time strings for Systems Manager

Many AWS Systems Manager API actions accept filters to limit the number of results returned by a request. Some of these API actions accept filters that require a formatted string to represent a specific date and time. For example, the DescribeSessions API action accepts the InvokedAfter and InvokedBefore keys as some of the valid values for a SessionFilter object. Another example is the DescribeAutomationExecutions API action, which accepts the StartTimeBefore and StartTimeAfter keys as some of the valid values for an AutomationExecutionFilter object. The values you provide for these keys when filtering your requests must match the ISO 8601 standard. For information about ISO 8601, see ISO 8601.

These formatted date and time strings are not limited to filters. There are also API actions that require an ISO 8601 formatted string to represent a specific date and time when providing a value for a request parameter. For example, the AtTime request parameter for the GetCalendarState action. These strings are difficult to create. Use the examples in this topic to create formatted date and time strings to use with Systems Manager API actions.

Formatting date and time strings for Systems Manager

The following is an example of an ISO 8601 formatted date and time string.

2020-05-08T15:16:43Z

This represents May 8th, 2020 at 15:16 Universal Coordinated Time (UTC). The calendar date portion of the string is represented by a four-digit year, two-digit month, and two-digit day separated by hyphens. This can be represented in the following format.

YYYY-MM-DD

The time portion of the string begins with the letter “T” as a delimiter, and then is represented by a two-digit hour, two-digit minute, and two-digit second separated by colons. This can be represented in the following format.

hh:mm:ss

The time portion of the string ends with the letter “Z”, denoting the UTC standard.
Creating custom date and time strings for Systems Manager

You can create custom date and time strings from your local machine using your preferred command line tool. The syntax you use to create an ISO 8601 formatted date and time string differs depending on your local machine's operating system. The following are examples of how you can use `date` from GNU's `coreutils` on Linux, or PowerShell on Windows to create an ISO 8601 formatted date and time string.

### coreutils

```bash
date '+%Y-%m-%dT%H:%M:%SZ'
```

### PowerShell

```powershell
(Get-Date).ToString("yyyy-MM-ddTH:mm:ssZ")
```

When working with Systems Manager API actions, you might need to create historical date and time strings for reporting or troubleshooting purposes. The following are examples of how you can create and use custom historical ISO 8601 formatted date and time strings for the AWS Tools for PowerShell and AWS CLI.

### AWS CLI

- Retrieve the last week of command history for an SSM document.

```bash
lastWeekStamp=$(date '+%Y-%m-%dT%H:%M:%SZ' -d '7 days ago')
docFilter='{"key":"DocumentName","value":"AWS-RunPatchBaseline"}'}
timeFilter='{"key":"InvokedAfter","value":'"$lastWeekStamp"'}'
commandFilters=[docFilter,timeFilter]
aws ssm list-commands 
   --filters $commandFilters
```

- Retrieve the last week of automation execution history.

```bash
lastWeekStamp=$(date '+%Y-%m-%dT%H:%M:%SZ' -d '7 days ago')
aws ssm describe-automation-executions 
   --filters Key=StartTimeAfter,Values=$lastWeekStamp
```

- Retrieve the last month of session history.

```bash
lastWeekStamp=$(date '+%Y-%m-%dT%H:%M:%SZ' -d '30 days ago')
aws ssm describe-sessions 
   --state History 
   --filters key=InvokedAfter,value=$lastWeekStamp
```

### AWS Tools for PowerShell

- Retrieve the last week of command history for an SSM document.

```powershell
$lastWeekStamp = (Get-Date).AddDays(-7).ToString("yyyy-MM-ddTH:mm:ssZ")
```
Creating custom date and time strings for Systems Manager

$docFilter = @{
    Key="DocumentName"
    Value="AWS-InstallWindowsUpdates"
}
$timeFilter = @{
    Key="InvokedAfter"
    Value=$lastWeekStamp
}
$commandFilters = $docFilter,$timeFilter
Get-SSMCommand ` `-Filters $commandFilters

- Retrieve the last week of automation execution history.

$lastWeekStamp = (Get-Date).AddDays(-7).ToString("yyyy-MM-ddTH:mm:ssZ")
Get-SSMAutomationExecutionList ` `-Filters @{Key="StartTimeAfter";Values=$lastWeekStamp}

- Retrieve the last month of session history.

$lastWeekStamp = (Get-Date).AddDays(-30).ToString("yyyy-MM-ddTH:mm:ssZ")
Get-SSMSession ` `-State History ` `-Filters @{Key="InvokedAfter";Value=$lastWeekStamp}
Use cases and best practices

This topic lists common use cases and best practices for AWS Systems Manager capabilities. If available, this topic also includes links to relevant blog posts and technical documentation.

Note
The title of each section here is an active link to the corresponding section in the technical documentation.

Automation (p. 278)
- Create self-service runbooks for infrastructure as Automation documents.
- Use Automation to simplify creating AMIs from the AWS Marketplace or custom AMIs, using public SSM documents or by authoring your own workflows.
- Build and maintain AMIs (p. 568) using the AWS-UpdateLinuxAmi and AWS-UpdateWindowsAmi Automation documents, or using custom Automation documents that you create.

Inventory (p. 704)
- Use Systems Manager Inventory with AWS Config to audit your application configurations over time.

Maintenance Windows (p. 620)
- Define a schedule to perform potentially disruptive actions on your instances such as OS patching, driver updates, or software installations.

Parameter Store (p. 198)
- Use Parameter Store to centrally manage global configuration settings.
- Use Parameter Store to encrypt and manage secrets by using AWS Key Management Service.
- Reference AWS Secrets Manager secrets from Parameter Store parameters (p. 98).
- Use Parameter Store with ECS task definitions to store secrets.

Patch Manager (p. 911)
- Use patch manager to roll out patches at scale and increase fleet compliance visibility across your instances.

Run Command (p. 829)
- Manage Instances at Scale without SSH Access Using EC2 Run Command.
- Audit all API calls made by on or on behalf of Run Command using AWS CloudTrail.
- Use the targets and rate control features in Run Command to perform a staged command execution (p. 840).
- Use fine-grained access permissions for Run Command (and all Systems Manager capabilities) by using AWS Identity and Access Management (IAM) policies (p. 1117).
State Manager (p. 864)

- Update SSM Agent at least once a month using the pre-configured AWS-UpdateSSMAgent document (p. 908).
- Bootstrap EC2 Instances on launch using EC2Config for Windows.
- (Windows) Upload the PowerShell or DSC module to Amazon S3, and use AWS-InstallPowerShellModule.
- Use Amazon EC2 tags to create application groups for your instances. And then target instances using the Targets parameter instead of specifying individual instance IDs.
- Automatically remediate findings generated by Amazon Inspector by using Systems Manager.
- Use a centralized configuration repository for all of your SSM documents, and share documents across your organization (p. 1091).

Managed instances (p. 758)

- Systems Manager requires accurate time references in order to perform its operations. If your instance's date and time are not set correctly, they may not match the signature date of your API requests. In some cases, this will lead to errors or incomplete functionality. For example, instances with incorrect time settings will not be included in your lists of managed instances.

For information on setting the time on your instances, see the following topics:

- Setting the Time for Your Linux Instance
- Setting the Time for a Windows Instance
## Document history

The following table describes the important changes to the documentation since the last release of AWS Systems Manager. For notification about updates to this documentation, you can subscribe to an RSS feed.

### Important

An updated version of SSM Agent is released whenever new capabilities are added to Systems Manager or updates are made to existing capabilities. If an older version of the agent is running on an instance, some SSM Agent processes can fail. For that reason, we recommend that you automate the process of keeping SSM Agent up-to-date on your instances. For information, see Automate updates to SSM Agent (p. 93). To be notified about SSM Agent updates, subscribe to the [SSM Agent Release Notes page on GitHub](https://github.com/aws-solutions/aws-ssm-agent).

- **API version:** 2014-11-06

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample scenario for using the <code>InstallOverrideList</code> parameter in AWS-RunPatchBaseline (p. 1201)</td>
<td>The new topic Sample scenario for using the InstallOverrideList parameter describes a strategy for using the InstallOverrideList parameter in the AWS-RunPatchBaseline document to apply different types of patches to a target group, on different maintenance window schedules, while still using a single patch baseline.</td>
<td>June 11, 2020</td>
</tr>
<tr>
<td>Predefined deployment strategies for AppConfig (p. 1201)</td>
<td>AppConfig now offers predefined deployment strategies. For more information, see Creating a deployment strategy.</td>
<td>June 10, 2020</td>
</tr>
<tr>
<td>Patch Manager now supports Red Hat Enterprise Linux (RHEL) 7.8-8.2 (p. 1201)</td>
<td>You can now use Patch Manager to patch RHEL 7.8 - 8.2 instances. For more information, see the following topics:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• How security patches are selected (p. 913)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• How patches are installed (p. 919)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• How patch baseline rules work on RHEL (p. 930)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Manually install SSM Agent on Red Hat Enterprise Linux instances (p. 79)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Install SSM Agent for a hybrid environment (Linux) (p. 52)</td>
<td>June 9, 2020</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Explorer supports delegated administration (p. 1201)</td>
<td>If you aggregate Explorer data from multiple AWS Regions and accounts by using resource data sync with AWS Organizations, then we suggest that you configure a delegated administrator for Explorer. A delegated administrator improves Explorer security by limiting the number of Explorer administrators who can create or delete multi-account and Region resource data syncs to only one individual. You also no longer need to be logged into the AWS Organizations master account to administer resource data syncs in Explorer. For more information, see Configuring a Delegated Administrator.</td>
<td>June 3, 2020</td>
</tr>
<tr>
<td>Apply State Manager association only at the next specified Cron interval (p. 1201)</td>
<td>If you don't want a State Manager association to run immediately after you create it, you can choose the Apply association only at the next specified Cron interval option in the Systems Manager console. For more information, see Create an association.</td>
<td>June 3, 2020</td>
</tr>
<tr>
<td>New data source in Explorer: AWS Compute Optimizer (p. 1201)</td>
<td>Explorer now displays data from AWS Compute Optimizer. This includes a count of Under provisioned and Over provisioned EC2 instances, optimization findings, on-demand pricing details, and recommendations for instance type and price. For more information, see the details for setting up AWS Compute Optimizer in Setting up related services.</td>
<td>May 26, 2020</td>
</tr>
<tr>
<td>New chapter: Tagging Systems Manager Resources (p. 1201)</td>
<td>The new chapter Tagging Systems Manager resources provides an overview of how you can use tags with the six taggable resource types in Systems Manager. The chapter also provides comprehensive instructions for adding and removing tags from these resource types:</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>• Documents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Maintenance windows</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Managed instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• OpsItems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Patch baselines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Windows Service Packs and Linux minor version upgrades using Patch Manager (p. 1201)</td>
<td>The new topic Walkthrough: Create a patch baseline for installing Windows Service Packs (console) demonstrates how you can create a patch baseline devoted exclusively to installing Windows Service Packs. The topic Create a custom patch baseline (Linux) has been updated with information about including minor version upgrades for Linux operating systems in patch baselines.</td>
<td></td>
</tr>
<tr>
<td>Parameter Store chapter reorganization (p. 1201)</td>
<td>All topics that deal with configuring or setting options for Parameter Store operations have been consolidated into the Getting started with Parameter Store section. This includes the topics Managing parameter tiers and Increasing Parameter Store throughput, which have been relocated from other parts of the chapter.</td>
<td></td>
</tr>
<tr>
<td>New topic for creating date and time strings for interacting with Systems Manager API actions. (p. 1201)</td>
<td>The new topic Creating formatted date and time strings for Systems Manager describes how to create formatted date and time strings for interacting with Systems Manager API actions.</td>
<td></td>
</tr>
</tbody>
</table>

May 25, 2020

May 21, 2020

May 18, 2020

May 13, 2020
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>About permissions for encrypting SecureString parameters (p. 1201)</td>
<td>The new topic IAM permissions for using AWS default keys and customer managed keys explains the difference between encrypting your SecureString parameters using a customer master key (CMK) and using the default AWS KMS key provided by AWS.</td>
<td>May 13, 2020</td>
</tr>
</tbody>
</table>
| Patch Manager now supports the Debian and Oracle Linux 7.6 operating systems (p. 1201) | You can now use Patch Manager to patch Debian and Oracle Linux instances. Patch Manager supports patching Debian 8.x and 9.x and Oracle Linux 7.6 versions. For more information, see the following topics:  
  - How security patches are selected (p. 913)  
  - How patches are installed (p. 919)  
  - How patch baseline rules work on Debian (p. 928)  
  - How patch baseline rules work on Oracle Linux (p. 928) | May 7, 2020 |
<p>| Create State Manager associations that target AWS Resource Groups (p. 1201) | In addition to targeting tags, individual instances, and all instances in your AWS account, you can now create State Manager associations that target instances in AWS Resource Groups. For more information, see About targets and rate controls in State Manager associations | May 7, 2020 |</p>
<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New aws:ec2:image data type in Parameter Store to validate AMI IDs</td>
<td>When you create a String parameter, you can now specify a data type as <code>aws:ec2:image</code> to ensure that the parameter value you enter is a valid Amazon Machine Image (AMI) ID format. Support for AMI ID formats lets you avoid updating all your scripts and templates with a new ID each time the AMI that you want to use in your processes changes. You can create a parameter with the data type <code>aws:ec2:image</code>, and for its value, enter the ID of an AMI. This is the AMI from which you currently want new instances to be created. You then reference this parameter in your templates, commands. When you're ready to use a different AMI, update the parameter value. Parameter Store validates the new AMI ID, and you don't need to update your scripts and templates. For more information, see Native parameter support for Amazon Machine Image IDs.</td>
<td>May 5, 2020</td>
</tr>
<tr>
<td>Managing exit codes in Run Command commands</td>
<td>Run Command lets you define how exit codes are handled in your scripts. By default, the exit code of the last command run in a script is reported as the exit code for the entire script. However, you can include a shell conditional statement to exit the script if any command before the final one fails using the following approach. For examples, see the new topic Managing exit codes in Run Command commands.</td>
<td>May 5, 2020</td>
</tr>
<tr>
<td>New public parameters released for availability zones and local zones</td>
<td>Public parameters have been released to make information about AWS availability zones and local zones available programmatically. These are in addition to existing global infrastructure public parameters for AWS services and Regions. For more information, see Calling AWS service, Region, endpoint, availability zone, and local zone public parameters.</td>
<td>May 4, 2020</td>
</tr>
<tr>
<td>New data source in Explorer: AWS Trusted Advisor (p. 1201)</td>
<td>Explorer now displays data from AWS Trusted Advisor. This includes the status of best practice checks and recommendations in the following areas: cost optimization, security, fault tolerance, performance, and service limits. For more information, see the details for setting up Trusted Advisor in Setting up related services.</td>
<td>May 4, 2020</td>
</tr>
<tr>
<td>New topic for AppConfig configuration store quotas and limitations (p. 1201)</td>
<td>We've added a new topic to help users quickly understand the service quotas and limitations of each AppConfig configuration store. For more information, see About configuration store quotas and limitations.</td>
<td>April 17, 2020</td>
</tr>
<tr>
<td>View Amazon CloudWatch alarms for AWS Resource Groups (p. 1201)</td>
<td>In addition to the four operations data tabs added to the AWS Resource Groups page in Systems Manager earlier this month, the AWS Resource Groups page now includes a Monitoring tab. This tab shows CloudWatch alarms and dashboards for the resources in the selected group. These operations data tabs are not available when viewing a group in the AWS Resource Groups service console. You can use the information on these tabs to help you understand which resources in a group are compliant and working correctly and which resources require action. If you need to take action on a resource, you can use Systems Manager Automation runbooks to perform common operations maintenance and troubleshooting tasks. For more information, see Viewing operations Data for AWS Resource Groups.</td>
<td>March 25, 2020</td>
</tr>
</tbody>
</table>
You can create State Manager associations that run Chef cookbooks and recipes by using the AWS-ApplyChefRecipes document. This document offers the following benefits for running Chef recipes:

- Supports multiple releases of Chef (Chef 11 through Chef 14).
- Automatically installs the Chef client software on target instances.
- Optionally runs Systems Manager compliance checks on target instances, and stores the results of compliance checks in an S3 bucket.
- Runs multiple cookbooks and recipes in a single run of the document.
- Optionally runs recipes in why-run mode, to show which recipes will change on target instances without making changes.
- Optionally applies custom JSON attributes to chef-client runs.

For more information, see Creating associations that run Chef recipes.

The topic AWS Resource Groups has been updated to provide more comprehensive information about working with resource groups in Systems Manager, and about using resource groups in your Systems Manager operations.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>View operations data for AWS Resource Groups</strong> (p. 1201)</td>
<td>The AWS Resource Groups page in Systems Manager now displays operations data for a selected group on four tabs: <strong>Details, Config, CloudTrail, OpsItems</strong>. These tabs are not available when viewing a group in the AWS Resource Groups service console. You can use the information on these tabs to help you understand which resources in a group are compliant and working correctly and which resources require action. If you need to take action on a resource, you can use Systems Manager Automation runbooks to perform common operations maintenance and troubleshooting tasks. For more information, see <strong>Viewing operations data for AWS Resource Groups</strong>.</td>
<td>March 16, 2020</td>
</tr>
<tr>
<td><strong>Synchronize inventory data from multiple AWS accounts to a central Amazon S3 bucket</strong> (p. 1201)</td>
<td>You can synchronize Systems Manager Inventory data from multiple AWS accounts to a central S3 bucket. The accounts must be defined in AWS Organizations. For more information, see <strong>Creating an Inventory Resource Data Sync for multiple accounts defined in AWS Organizations</strong>.</td>
<td>March 16, 2020</td>
</tr>
<tr>
<td><strong>Store AppConfig configurations in Amazon S3</strong> (p. 1201)</td>
<td>Previously, AppConfig only supported application configurations that were stored in Systems Manager (SSM) documents or Parameter Store parameters. In addition to these options, AppConfig now supports storing configurations in Amazon S3. For more information, see <strong>About configurations stored in Amazon S3</strong>.</td>
<td>March 13, 2020</td>
</tr>
<tr>
<td><strong>SSM Agent installed by default on Amazon ECS-optimized AMIs</strong> (p. 1201)</td>
<td>SSM Agent is now installed by default on Amazon ECS-Optimized AMIs. For more information, see <strong>Working with SSM Agent</strong>.</td>
<td>February 25, 2020</td>
</tr>
<tr>
<td>Change Description</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>AppConfig now enables you to create an application configuration in the console at the time you create a configuration profile. For more information, see Create a Configuration and a configuration Profile.</td>
<td>February 13, 2020</td>
<td></td>
</tr>
<tr>
<td>In addition to the option for automatically approving patches for installation a specified number of days after they are released, Patch Manager now supports the ability to auto-approve only patches released on or before a date that you specify. For example, if you specify July 7, 2020, as the cutoff date in your patch baseline, no patches released on or after July 8, 2020, are installed automatically. For more information, see About custom baselines and Create a custom patch baseline.</td>
<td>February 12, 2020</td>
<td></td>
</tr>
</tbody>
</table>
| When you register a maintenance window task, you specify the parameters that are unique to the task type. You can reference certain values using pseudo parameter syntax, such as {{TARGET_ID}}, {{TARGET_TYPE}}, and {{WINDOW_TARGET_ID}}. When the maintenance window task runs, it passes the correct values instead of the pseudo parameter placeholders. To support resources that are part of a resource group as a target, you can use the {{RESOURCE_ID}} pseudo parameter to pass values for resources such as DynamoDB tables, S3 buckets, and other supported types. For more information, see the following topics in Tutorial: Create and configure a maintenance window (AWS CLI):  
  - About pseudo parameters  
  - Examples: Register tasks with a maintenance window | February 6, 2020 |
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quickly rerun commands</td>
<td>Systems Manager includes two options to help you rerun a command from the Run Command page in the AWS Systems Manager console. <strong>Rerun:</strong> This button enables you to run the same command without making changes to it. <strong>Copy to new:</strong> This button copies the settings of one command to a new command and gives you the option to edit those settings before you run it. For more information, see Rerunning commands.</td>
<td>February 5, 2020</td>
</tr>
<tr>
<td>Reverting from the advanced-instances tier to the standard-instances tier</td>
<td>If you previously configured all on-premises instances running in your hybrid environment to use the advanced-instances tier, you can now quickly configure those instances to use the standard-instance tier. Reverting to the standard-instances tier applies to all hybrid instances in an AWS account and a single AWS Region. Reverting to the standard-instances tier impacts the availability of some Systems Manager capabilities. For more information, see Reverting from the advanced-instances tier to the standard-instances tier.</td>
<td>January 16, 2020</td>
</tr>
<tr>
<td>New option to skip instance reboots after patch installation</td>
<td>Previously, managed instances were always rebooted after Patch Manager installed patches on them. A new RebootOption parameter in the SSM document AWS-RunPatchBaseline lets you specify whether or not you want your instances to reboot automatically after new patches are installed. For more information, see Parameter name: RebootOption in the topic About the SSM document AWS-RunPatchBaseline.</td>
<td>January 15, 2020</td>
</tr>
<tr>
<td>New topic: ‘Running PowerShell scripts on Linux instances’</td>
<td>A new topic that describes how to use Run Command to run PowerShell scripts on Linux instances. For more information, see Running PowerShell scripts on Linux instances.</td>
<td>January 10, 2020</td>
</tr>
<tr>
<td>New topic: ‘Automation document schema and syntax’ (p. 1201)</td>
<td>Examples and information about the elements in Schema 0.3, which is used by SSM Automation documents, can now be found in the topic Automation document schema and syntax.</td>
<td>January 10, 2020</td>
</tr>
<tr>
<td>Updates to ‘configure SSM Agent to use a proxy’ (p. 1201)</td>
<td>The values to specify when configuring SSM Agent to use a proxy have been updated to reflect options for both HTTP proxy servers and HTTPS proxy servers. For more information, see Configure SSM Agent to use a proxy.</td>
<td>January 9, 2020</td>
</tr>
<tr>
<td>New “Security” chapter outlines practices for securing Systems Manager resources (p. 1201)</td>
<td>A new Security chapter in the AWS Systems Manager User Guide helps you understand how to apply the shared responsibility model when using Systems Manager. Topics in the chapter show you how to configure Systems Manager to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Systems Manager resources.</td>
<td></td>
</tr>
</tbody>
</table>

**Note**

As part of this update, the user guide chapter "Authentication and Access Control" has been replaced by a new, simpler section, Identity and access management for AWS Systems Manager.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New sample custom Automation documents (p. 1201)</td>
<td>A set of sample custom Automation documents has been added to the user guide. These samples show how to use various Automation actions to simplify deployment, troubleshooting, and maintenance tasks, and are intended to help you write your own custom Automation documents. For more information, see Custom Automation document samples. You can also view Amazon Managed Automation Document content in the Systems Manager Console. For more information, see Systems Manager Automation document details reference.</td>
<td>December 23, 2019</td>
</tr>
<tr>
<td>Support for the Oracle Linux (p. 1201)</td>
<td>Systems Manager now supports Oracle Linux 7.5 and 7.7. For information about manually installing SSM Agent on EC2 instances for Oracle Linux instances, see Oracle Linux. For information about installing SSM Agent on Oracle Linux servers in a hybrid environment, see Step 6: Install SSM Agent for a hybrid environment (Linux).</td>
<td>December 19, 2019</td>
</tr>
<tr>
<td>Add a script from another Automation document to your workflow (p. 1201)</td>
<td>When you create an Automation document using a command line tool, you can now add a script that is already used in another document to a step in your new document. (Scripts are used with the aws:executeScript action type.) You can add scripts from Automation documents that you own or that are shared with you from another AWS account. For details, see Creating an Automation document that runs scripts (command line).</td>
<td>December 19, 2019</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>Launch Session Manager sessions from the Amazon EC2 console (p. 1201)</strong></td>
<td>You can now start Session Manager sessions from the Amazon Elastic Compute Cloud (Amazon EC2) console. Working with session-related tasks from the Amazon EC2 console requires different IAM permissions for both users and administrators. You can provide permissions for using the Session Manager console and AWS CLI only, for using the Amazon EC2 console only, or for using all three tools. For more information, see the following topics.</td>
<td>December 18, 2019</td>
</tr>
<tr>
<td><strong>• Quickstart default IAM policies for Session Manager</strong>&lt;br&gt;<strong>• Starting a session (Amazon EC2 console)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CloudWatch support for Run Command metrics and alarms (p. 1201)</strong></td>
<td>AWS Systems Manager now publishes metrics about the status of Run Command commands to CloudWatch, enabling you to set alarms based on those metrics. The terminal status values for commands for which you can track metrics include <strong>Success</strong>, <strong>Failed</strong>, and <strong>Delivery Timed Out</strong>. For more information, see <strong>Monitoring Run Command metrics using Amazon CloudWatch</strong>.</td>
<td>December 17, 2019</td>
</tr>
<tr>
<td><strong>New Systems Manager capability: Change Calendar (p. 1201)</strong></td>
<td>Use Systems Manager Change Calendar to specify periods of time (events) during which you want to limit or prevent code changes (such as from Systems Manager Automation documents or AWS Lambda functions) to resources. A Change Calendar is a new Systems Manager document type that stores <strong>iCalendar 2.0</strong> data in plaintext format. For more information, see <strong>AWS Systems Manager Change Calendar</strong>.</td>
<td>December 11, 2019</td>
</tr>
<tr>
<td>New Systems Manager capability: AWS AppConfig (p. 1201)</td>
<td>Use AppConfig to create, manage, and quickly deploy application configurations. AppConfig supports controlled deployments to applications of any size. You can use AppConfig with applications hosted on EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices. To prevent errors when deploying application configurations, AppConfig includes validators. A validator provides a syntactic or semantic check to ensure that the configuration you want to deploy works as intended. During a configuration deployment, AppConfig monitors the application to ensure that the deployment is successful. If the system encounters an error or if the deployment triggers an alarm, AppConfig rolls back the change to minimize impact for your application users. For more information, see AWS AppConfig.</td>
<td>November 25, 2019</td>
</tr>
<tr>
<td>Changes to left navigation in Systems Manager console (p. 1201)</td>
<td>You can now find AWS Resource Groups and Parameter Store under Application Management. To view the documentation for these capabilities, see AWS Systems Manager Application Management.</td>
<td>November 25, 2019</td>
</tr>
</tbody>
</table>
AWS Systems Manager Explorer is a customizable operations dashboard that reports information about your AWS resources. Explorer displays an aggregated view of operations data (OpsData) for your AWS accounts and across Regions. In Explorer, OpsData includes metadata about your EC2 instances, patch compliance details, and operational work items (OpsItems). Explorer provides context about how OpsItems are distributed across your business units or applications, how they trend over time, and how they vary by category. You can group and filter information in Explorer to focus on items that are relevant to you and that require action. When you identify high priority issues, you can use Systems Manager OpsCenter to run Automation runbooks and quickly resolve those issues. For information see, AWS Systems Manager Explorer.

**Note**
Set up for Systems Manager OpsCenter is integrated with set up for Explorer. If you already set up OpsCenter, you still need to complete Integrated Setup to verify settings and options. If you have not set up OpsCenter, then you can use Integrated Setup to get started with both capabilities. For more information, see Getting started with Explorer and OpsCenter.
Improved parameter search capabilities (p. 1201)  

The tools for searching for parameters now make it easier to find parameters when you have a large number of them in your account or when you do not remember the exact name of a parameter. The search tool now lets you filter by contains. Previously, the search tool supported searching for parameter names only by equals and begins-with. For more information, see Searching for Systems Manager parameters.

New console-based Document Builder for Automation | Support for running scripts in Automation steps (p. 1201)  

You can now use Systems Manager Automation to build and share standardized operational playbooks to ensure consistency across users, accounts, and AWS Regions. With this ability to run scripts and add inline documentation to your Automation documents using Markdown, you can reduce errors and eliminate manual steps such as navigating written procedures in wikis and running terminal commands.

For more information, see the following topics.

- Walkthrough: Using Document Builder to create a custom Automation document
- Amazon managed Automation documents that run scripts
- aws:executeScript (Automation actions reference)
- Creating Automation documents that run scripts
- Creating Automation documents using Document Builder
- New Automation Features In Systems Manager on the AWS News Blog

November 15, 2019

November 14, 2019
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
</table>
| Perform an in-place package update using Distributor        | Previously, when you wanted to install an update to a package using Distributor, your only choice was to uninstall the entire package and reinstall the new version. Now you can choose to perform an in-place update instead. During an in-place update, Distributor installs only files that are new or changed since the last installation, according to the update script you include in your package. With this option, your package application can remain available and not be taken offline during the update. For more information, see the following topics.  
  - Create a package  
  - Install or update packages                                      | November 11, 2019 |
<p>| New SSM Agent auto update feature                            | With one click, you can configure all instances in your AWS account to automatically check for and download new versions of SSM Agent. To do this, choose Agent auto update on the Managed instances page in the AWS Systems Manager console. For information, see Automate updates to SSM Agent. | November 5, 2019 |
| Restrict Session Manager access using AWS-supplied tags      | A second method for controlling user access to session actions is now available. This new method lets you create IAM access policies using AWS-supplied session tags instead of using the <code>{aws:username}</code> variable. Using these AWS-supplied session tags makes it possible for organizations that use federated IDs to control user access to sessions. For information, see Allow a user to terminate only sessions they started. | October 2, 2019 |</p>
<table>
<thead>
<tr>
<th>New Ansible SSM document: AWS-ApplyAnsiblePlaybooks (p. 1201)</th>
<th>You can create State Manager associations that run Ansible Playbooks by using the AWS-ApplyAnsiblePlaybooks document. This document offers the following benefits for running Playbooks:</th>
</tr>
</thead>
</table>
|  | • Support for running complex Playbooks  
• Support for downloading Playbooks from GitHub and Amazon Simple Storage Service (Amazon S3)  
• Support for compressed Playbook structure  
• Enhanced logging  
• Ability to specify which Playbook to run when Playbooks are bundled |
|  | For more information, see Creating associations that run Ansible playbooks |
| Port forwarding support for Session Manager (p. 1201) | Session Manager now supports port forwarding sessions. Port forwarding allows you to securely create tunnels between your instances deployed in private subnets, without the need to start the SSH service on the server, to open the SSH port in the security group, or to use a bastion host. Similar to SSH tunnels, port forwarding allows you to forward traffic between your laptop to open ports on your instance. Once port forwarding is configured, you can connect to the local port and access the server application running inside the instance. For more information, see the following topics: |
|  | • Port Forwarding Using AWS Systems Manager Session Manager on the AWS News Blog  
• Starting a session (port forwarding) |
|  | September 24, 2019  
August 29, 2019 |
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify a default parameter tier or automate tier selection (p. 1201)</td>
<td>You can now specify a default parameter tier to use for requests to create or update a parameter that do not specify a tier. You can set the default tier to standard parameters, advanced parameters, or a new option, Intelligent-Tiering. Intelligent-Tiering evaluates each PutParameter request and creates an advanced parameter only when required. (Advanced parameters are required if the size of the parameter value is more than 4 KB, a parameter policy is associated with the parameter, or the maximum 10,000 parameters supported for the standard tier are already created.) For more information about specifying a default tier and using Intelligent-Tiering, see Specifying a default parameter tier.</td>
<td>August 27, 2019</td>
</tr>
<tr>
<td>Working with associations section updated with CLI and PowerShell procedures (p. 1201)</td>
<td>The Working with Associations section has been updated to include procedural documentation for managing associations using the AWS CLI or AWS Tools for PowerShell. For information see, Working with associations in Systems Manager.</td>
<td>August 26, 2019</td>
</tr>
<tr>
<td>Working with Automation executions section updated with CLI and PowerShell procedures (p. 1201)</td>
<td>The Working with Automation Executions section has been updated to include procedural documentation for running Automation workflows using the AWS CLI or AWS Tools for PowerShell. For information see, Working with Automation executions.</td>
<td>August 20, 2019</td>
</tr>
<tr>
<td>OpsCenter integrates with application insights (p. 1201)</td>
<td>OpsCenter integrates with Amazon CloudWatch Application Insights for .NET and SQL Server. This means you can automatically create OpsItems for problems detected in your applications. For information about how to configure Application Insights to create OpsItems, see Setting up your application in the Amazon CloudWatch User Guide.</td>
<td>August 7, 2019</td>
</tr>
</tbody>
</table>
New console feature: AWS Systems Manager Quick Setup (p. 1201)

Quick Setup is a new feature in the Systems Manager console that helps you quickly configure several Systems Manager components on your EC2 instances. Specifically, Quick Setup helps you configure the following components on the instances you choose or target by using tags:

- An AWS Identity and Access Management (IAM) instance profile role for Systems Manager.
- A scheduled, bi-monthly update of SSM Agent.
- A scheduled collection of Inventory metadata every 30 minutes.
- A daily scan of your instances to identify missing patches.
- A one-time installation and configuration of the Amazon CloudWatch agent.
- A scheduled, monthly update of the CloudWatch agent.

For more information, see AWS Systems Manager Quick Setup.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
</table>
| Register a resource group as a maintenance window target (p. 1201) | In addition to registering managed instances as the target of a maintenance window, you can now register a resource group as a maintenance window target. Maintenance Windows supports all the AWS resource types that are supported by AWS Resource Groups including AWS::EC2::Instance, AWS::DynamoDB::Table, AWS::OpsWorks::Instance, AWS::Redshift::Cluster, and more. With this release you can also send commands to a resource group, for example by using the Run Command console or the AWS CLI `send-command` command. For more information, see the following topics:  
  - Assign targets to a maintenance window (console)
  - Examples: Register targets with a maintenance window
  - Using targets and rate controls to send commands to a fleet | July 23, 2019 |
<p>| Simplified package creation and versioning with AWS Systems Manager Distributor (p. 1201) | Distributor has a new, simplified package creation workflow that can generate a package manifest, scripts, and file hashes for you. You can also use the simplified workflow when you add a version to an existing package. | July 22, 2019 |
| New document categories pane for Systems Manager Automation (p. 1201) | Systems Manager includes a new Document categories pane when you run an Automation in the console. Use this pane to filter Automation documents based on their purpose. | July 18, 2019 |</p>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify user permissions to access the default Session Manager configuration document (p. 1201)</td>
<td>When a user in your account uses the AWS CLI to start a Session Manager session and doesn't specify a configuration document in the command, Systems Manager uses the default configuration document <code>SSM-SessionManagerRunShell</code>. You can now verify that the user has been granted permission to access this document by adding a condition element for <code>ssm:SessionDocumentAccessCheck</code> to the IAM user's policy. For information, see Enforce document permission check for default CLI scenario.</td>
<td>July 9, 2019</td>
</tr>
<tr>
<td>Support for starting Session Manager sessions using operating system user credentials (p. 1201)</td>
<td>By default, Session Manager sessions are launched using the credentials of a system-generated <code>ssm-user</code> account that is created on a managed instance. On Linux machines, you can now instead launch sessions using the credentials of an operating system account. For information, see Enable Run As support for Linux instances.</td>
<td>July 9, 2019</td>
</tr>
<tr>
<td>Support for starting Session Manager sessions using SSH (p. 1201)</td>
<td>You can now use the AWS CLI to start an SSH session on a managed instance using Session Manager. For information about enabling SSH sessions with Session Manager, see (Optional) Enable SSH Session Manager sessions. For information about starting an SSH session using Session Manager, see Starting a session (SSH).</td>
<td>July 9, 2019</td>
</tr>
<tr>
<td>Support for changing passwords on managed instances (p. 1201)</td>
<td>You can now reset passwords on machines that you manage using Systems Manager (managed instances). You can reset the password using the Systems Manager console or the AWS CLI. For information, see Resetting passwords on managed instances.</td>
<td>July 9, 2019</td>
</tr>
<tr>
<td>Revisions to “What is AWS Systems Manager?” (p. 1201)</td>
<td>The introductory content in What is AWS Systems Manager? has been expanded to provide a broader introduction to the service and reflect Systems Manager capabilities that have been released recently. In addition, other content in the section has been moved into individual topics for better discoverability.</td>
<td>June 10, 2019</td>
</tr>
<tr>
<td>New Systems Manager capability: OpsCenter (p. 1201)</td>
<td>OpsCenter provides a central location where operations engineers and IT professionals can view, investigate, and resolve operational work items (OpsItems) related to AWS resources. OpsCenter is designed to reduce mean time to resolution for issues impacting AWS resources. This Systems Manager capability aggregates and standardizes OpsItems across services while providing contextual investigation data about each OpsItem, related OpsItems, and related resources. OpsCenter also provides Systems Manager Automation documents (runbooks) that you can use to quickly resolve issues. You can specify searchable, custom data for each OpsItem. You can also view automatically-generated summary reports about OpsItems by status and source. For more information, see AWS Systems Manager OpsCenter.</td>
<td>June 6, 2019</td>
</tr>
<tr>
<td>Changes to Systems Manager left navigation pane in the AWS Management Console (p. 1201)</td>
<td>The Systems Manager left navigation pane in the AWS Management Console includes new headings, including a new heading for Ops Center, that provide a more logical grouping of Systems Manager capabilities.</td>
<td>June 6, 2019</td>
</tr>
<tr>
<td>Revised tutorial for creating and configuring a maintenance window using the AWS CLI (p. 1201)</td>
<td>Tutorial: Create and configure a maintenance window (AWS CLI) has been overhauled to provide a simple path through the practice steps. You create a single maintenance window, identify a single target, and set up a simple task for the maintenance window to run. Along the way, we provide information and examples you can use to create your own task registration commands, including information for using pseudo parameters such as {{TARGET_ID}}. For additional information and examples, see the following topics:</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Notifications about SSM Agent updates (p. 1201)</td>
<td>To be notified about SSM Agent updates, subscribe to the SSM Agent Release Notes page on GitHub.</td>
<td></td>
</tr>
<tr>
<td>Receive notifications or trigger actions based on changes in Parameter Store (p. 1201)</td>
<td>The topic Set up notifications or trigger actions based on Parameter Store events now helps you set up Amazon CloudWatch Events rules to respond to changes in Parameter Store. You can receive notifications or trigger other actions when any of the following occur:</td>
<td></td>
</tr>
</tbody>
</table>
|  | • A parameter is created, updated, or deleted.  
• A parameter label version is created, updated, or deleted.  
• A parameter expires, is going to expire, or hasn't changed in a specified period of time. |
| May 31, 2019 |  |
| May 24, 2019 |  |
| May 22, 2019 |  |

1224
We have expanded and reorganized the Setting Up and Getting Started content in the AWS Systems Manager User Guide. Setting Up content has been divided into two sections. One section focuses on tasks for setting up Systems Manager to configure and manage your EC2 instances. The other focuses on tasks for setting up Systems Manager to configure and manage your on-premises servers and virtual machines (VMs) in a hybrid environment. Both sections now present all setup topics as major numbered steps, in the recommended order of completion. A new Getting Started chapter focuses on helping end-users get started with Systems Manager after account and service configuration tasks have been completed.

- Setting up AWS Systems Manager
- Setting up AWS Systems Manager for hybrid environments
- Getting started with AWS Systems Manager
| Include patches for Microsoft applications in patch baselines (Windows) (p. 1201) | Patch Manager now supports patch updates for Microsoft applications on Windows Server instances. Previously, only patches for the Windows Server operating system were supported. Patch Manager provides two predefined patch baselines for Windows Server instances. The patch baseline `AWS-WindowsPredefinedPatchBaseline-OS` applies to operating system patches only. `AWS-WindowsPredefinedPatchBaseline-OS-Applications` applies to both the Windows Server operating system and Microsoft applications on Windows. For information about creating a custom patch baseline that includes patches for Microsoft applications, see the first procedure in Create a custom patch baseline. Also, as part of this update, the names of AWS-provided predefined patch baselines are being changed. For more information, see Predefined baselines. | May 7, 2019 |
| Examples for registering maintenance window targets using the AWS CLI (p. 1201) | The new topic Examples: Register targets with a maintenance window provides three sample commands to demonstrate different ways you can specify the targets for a maintenance window when you use the AWS CLI. The topic also explains the best use case for each of the sample commands. | May 3, 2019 |
| Updates to patch group topics (p. 1201) | The topic About patch groups has been updated to include a section on how managed instances determine the appropriate patch baseline to use during patching operations. Additionally, instructions have been added for using the AWS CLI or Systems Manager console to add Patch Group tags to your managed instances and how to add a Patch Group to a patch baseline. For more information see Create a patch group and Add a patch group to a patch baseline. | May 1, 2019 |
New Parameter Store features (p. 1201) Parameter Store offers the following new features:

- **Advanced parameters:** Parameter Store now enables you to individually configure parameters to use either a standard-parameter tier (the default tier) or an advanced-parameter tier. Advanced parameters offer a larger size limit for the parameter value, a higher limit for the number of parameters you can create per account and Region, and the ability to use parameter policies. For more information about advanced parameters, see About Systems Manager advanced parameters.

- **Parameter policies:** Parameter policies help you manage a growing set of parameters by enabling you to assign specific criteria to a parameter, such as an expiration date or time to live. Parameter policies are especially helpful in forcing you to update or delete passwords and configuration data stored in Parameter Store. Parameter policies are only available for parameters that use the advanced-parameter tier. For more information, see Working with parameter policies.

- **Higher throughput:** You can now increase the Parameter Store throughput limit to a maximum of 1,000 transactions per second. For more information, see Increasing Parameter Store throughput.
<table>
<thead>
<tr>
<th>Updates to the Automation section (p. 1201)</th>
<th>The Automation section has been updated for improved discoverability. In addition, three new topics have been added to the Automation section:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running an Automation workflow manually (p. 290)</td>
<td></td>
</tr>
<tr>
<td>Running an Automation workflow with approvers (p. 296)</td>
<td></td>
</tr>
<tr>
<td>Running Automation workflows based on triggers (p. 314)</td>
<td></td>
</tr>
<tr>
<td>Encrypt session data using an AWS KMS key (p. 1201)</td>
<td>By default, Session Manager uses TLS 1.2 to encrypt session data transmitted between the local machines of users in your account and your EC2 instances. Now you can choose to further encrypt that data using a customer master key (CMK) that has been created in AWS Key Management Service. You can use a key that has been created in your AWS account or one that has been shared with you from another account. For information about specifying a CMK to encrypt session data, see Enable AWS KMS key encryption of session data (console), Create Session Manager preferences (AWS CLI), or Update Session Manager preferences (AWS CLI).</td>
</tr>
<tr>
<td>Configuring Amazon SNS notifications for AWS Systems Manager (p. 1201)</td>
<td>Added instructions for using the AWS CLI or Systems Manager console to configure Amazon SNS notifications for Run Command and Run Command tasks registered to a maintenance window. For more information see Configuring Amazon SNS notifications for AWS Systems Manager.</td>
</tr>
<tr>
<td></td>
<td>April 17, 2019</td>
</tr>
<tr>
<td></td>
<td>April 4, 2019</td>
</tr>
<tr>
<td></td>
<td>March 6, 2019</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Advanced instances for servers and VMs in hybrid environments (p. 1201)</td>
<td>AWS Systems Manager offers a standard-instances tier and an advanced-instances tier for servers and VMs in your hybrid environment. The standard-instances tier enables you to register a maximum of 1,000 servers or VMs per AWS account per AWS Region. If you need to register more than 1,000 servers or VMs in a single account and Region, then use the advanced-instances tier. You can create as many instances as you like in the advanced-instances tier, but all instances configured for Systems Manager are available on a pay-per-use basis. Advanced instances also enable you to connect to your hybrid machines by using AWS Systems Manager Session Manager. Session Manager provides interactive shell access to your instances. For more information about enabling advanced instances, see Using the advanced-instances tier.</td>
</tr>
<tr>
<td>Create State Manager associations that use shared SSM documents (p. 1201)</td>
<td>You can create State Manager associations that use SSM Command and Automation documents shared from other AWS accounts. Creating associations by using shared SSM documents helps to keep your Amazon EC2 and hybrid infrastructure in a consistent state even when instances are not in the same account. For information about sharing SSM documents, see AWS Systems Manager Documents. For information about creating a State Manager association, see Create an association.</td>
</tr>
<tr>
<td>View lists of Systems Manager events supported for Amazon CloudWatch Events rules (p. 1201)</td>
<td>The new topic Monitoring Systems Manager events with Amazon CloudWatch Events provides a summary of the various events emitted by Systems Manager for which you can set up event monitoring rules in CloudWatch Events.</td>
</tr>
<tr>
<td><strong>Add tags when you create Systems Manager resources (p. 1201)</strong></td>
<td>Systems Manager now supports the ability to add tags to certain resource types when you create them. The resources you can tag when you create them with the AWS CLI or an SDK include maintenance windows, patch baselines, Parameter Store parameters, and SSM documents. You can also assign tags to a managed instance when you create an activation for it. When you use the Systems Manager console, you can add tags to maintenance windows, patch baselines, and parameters.</td>
</tr>
<tr>
<td><strong>Automatic IAM role creation for Systems Manager Inventory (p. 1201)</strong></td>
<td>Previously you had to create an AWS Identity and Access Management (IAM) role and attach separate policies to this role to view inventory data on the <strong>Inventory Detail View</strong> page in the console. You no longer need to create this role or attach policies to it. When you choose a Remote Data Sync on the <strong>Inventory Detail View</strong> page, Systems Manager automatically creates the <strong>Amazon-GlueServicePolicyForSSM</strong> role and assigns the <strong>Amazon-GlueServicePolicyForSSM-{{S3 bucket name}}</strong> policy and the <strong>AWSGlueServiceRole</strong> policy to it. For more information, see <strong>Querying inventory data from multiple Regions and accounts.</strong></td>
</tr>
<tr>
<td><strong>Maintenance Windows walkthroughs to update SSM Agent (p. 1201)</strong></td>
<td>Added two new walkthroughs to the Maintenance Windows documentation. The walkthroughs detail how to use the Systems Manager console or the AWS CLI to create a maintenance window that keeps SSM Agent up-to-date automatically. For more information, see <strong>Maintenance Windows walkthroughs.</strong></td>
</tr>
<tr>
<td><strong>Using Parameter Store public parameters (p. 1201)</strong></td>
<td>Added short section describing Parameter Store public parameters. For more information, see <strong>Using Systems Manager public parameters.</strong></td>
</tr>
<tr>
<td>Topic</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Use the AWS CLI to create Session Manager preferences (p. 1201)</td>
<td>Added instructions for using the AWS CLI to create Session Manager preferences, such as CloudWatch Logs, S3 bucket logging options, and session encryption settings. For more information, see Use the AWS CLI to create Session Manager preferences.</td>
</tr>
<tr>
<td>Executing Systems Manager automation workflows by using State Manager (p. 1201)</td>
<td>AWS Systems Manager State Manager now supports creating associations that use SSM Automation documents. State Manager previously supported only command and policy documents, which meant that you could only create associations that targeted managed instances. With support for SSM Automation documents, you can now create associations that target different types of AWS resources. For more information, see Executing Systems Manager Automation workflows by using State Manager.</td>
</tr>
<tr>
<td>Reference updates for cron and rate expressions and maintenance window scheduling options (p. 1201)</td>
<td>The reference topic Cron and rate expressions for Systems Manager has been revised. The new version provides more examples and improved explanations of how to use cron and rate expressions to schedule your maintenance windows and State Manager associations. In addition, the new topic Maintenance Windows scheduling and active period options explains how the various schedule-related options for maintenance windows (Start date, End date, Time zone, Schedule frequency) relate to one another.</td>
</tr>
<tr>
<td>Updates to the Systems Manager prerequisites topic (p. 1201)</td>
<td>The Systems Manager prerequisites topic has been updated to provide information about supported operating system versions in a more detailed tabular format, along with other changes in the page for improved readability.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Enable SSM Agent debug logging (p. 1201)</td>
<td>You can enable SSM Agent debug logging by editing the seelog.xml.template file on the managed instance. For more information, see Enable SSM Agent debug logging.</td>
</tr>
<tr>
<td>Support for ARM64 processor architectures (p. 1201)</td>
<td>AWS Systems Manager now supports ARM64 versions of the Amazon Linux 2, Red Hat Enterprise Linux 7.6, and Ubuntu Server (18.04 LTS and 16.04 LTS) operating systems. For more information, see the instructions for installing Amazon Linux 2, RHEL, and Ubuntu Server 18.04 and 16.04 LTS with Snap packages. For more information about the A1 instance type, see General purpose instances in the Amazon EC2 User Guide for Linux Instances.</td>
</tr>
<tr>
<td>Create and deploy packages by using AWS Systems Manager Distributor (p. 1201)</td>
<td>AWS Systems Manager Distributor lets you package your own software—or find AWS-provided agent software packages, such as AmazonCloudWatchAgent—to install on AWS Systems Manager managed instances. Distributor publishes resources, such as software packages, to AWS Systems Manager managed instances. Publishing a package advertises specific versions of the package's document—a Systems Manager document that you create when you add the package in Distributor—to managed instances that you identify by managed instance IDs, AWS account IDs, tags, or an AWS Region. For more information, see AWS Systems Manager Distributor.</td>
</tr>
<tr>
<td>Section Title</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Concurrently run AWS Systems Manager Automation workflows across multiple AWS Regions and AWS accounts from a central account (p. 1201)</td>
<td>You can concurrently run AWS Systems Manager automation workflows across multiple AWS Regions and AWS accounts or AWS Organizational Units (OUs) from an Automation management account. Concurrently executing Automations in multiple Regions and accounts or OUs reduces the time required to administer your AWS resources while enhancing the security of your computing environment. For more information see Executing Automation workflows in multiple AWS Regions and accounts.</td>
</tr>
<tr>
<td>Query inventory data from multiple AWS Regions and accounts (p. 1201)</td>
<td>Systems Manager Inventory integrates with Amazon Athena to help you query inventory data from multiple AWS Regions and accounts. Athena integration uses Resource Data Sync so that you can view inventory data from all of your managed instances on the Inventory Detail View page in the AWS Systems Manager console. For more information see Querying Inventory data from multiple Regions and accounts.</td>
</tr>
<tr>
<td>Create State Manager associations that run MOF files (p. 1201)</td>
<td>You can run Managed Object Format (MOF) files to enforce a desired state on Windows Server managed instances with State Manager by using the AWS-ApplyDSCMofs SSM document. The AWS-ApplyDSCMofs document has two execution modes. With the first mode, you can configure the association to scan and report if the managed instances are currently in the desired state defined in the specified MOF files. In the second mode, you can run the MOF files and change the configuration of your instances based on the resources and their values defined in the MOF files. The AWS-ApplyDSCMofs document enables you to download and run MOF configuration files from Amazon Simple Storage Service (Amazon S3), a local share, or from a secure web site with an HTTPS domain. For more information, see Creating associations that run MOF files.</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Restrict administrative access in Session Manager sessions (p. 1201)</td>
<td>Session Manager sessions are launched using the credentials of a user account that is created with default root or administrator privileges called ssm-user. Information about restricting administrative control for this account is now available in the topic Disable or enable ssm-user account administrative permissions.</td>
</tr>
<tr>
<td>YAML examples in Automation actions reference (p. 1201)</td>
<td>The Automations actions reference now includes a YAML sample for each action that already includes a JSON sample.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Assign compliance severity levels to associations (p. 1201)</td>
<td>You can now assign compliance severity levels to State Manager associations. These severity levels are reported in the Compliance Dashboard and can also be used to filter your compliance reports. The severity levels you can assign include Critical, High, Medium, Low, and Unspecified. For more information, see Create an association (console).</td>
</tr>
<tr>
<td>Use targets and rate controls with Automation and State Manager (p. 1201)</td>
<td>Control the execution of Automations and State Manager associations across your fleet of resources by using targets, concurrency, and error thresholds. For more information see Using targets and rate controls to run Automation workflows on a fleet and Using targets and rate controls with State Manager associations.</td>
</tr>
<tr>
<td>Specify active time ranges and international time zones for maintenance windows (p. 1201)</td>
<td>You can also specify dates that a maintenance window should not run before or after (start date and end date), and you can specify the international time zone on which to base the maintenance window schedule. For more information see Create a maintenance window (console) and Update a maintenance window (AWS CLI).</td>
</tr>
<tr>
<td>Maintain a custom list of patches for your patch baseline in an S3 bucket (p. 1201)</td>
<td>The new 'InstallOverrideList' parameter in the SSM document 'AWS-RunPatchBaseline' lets you specify an https URL or an Amazon Simple Storage Service (Amazon S3) path-style URL to a list of patches to be installed. This patch installation list, which you maintain in an S3 bucket in YAML format, overrides the patches specified by the default patch baseline. For more information, see Parameter name: InstallOverrideList.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Expanded control over whether patch dependencies are installed</td>
<td>Previously, if a patch in your Rejected patches list was identified as a dependency of another patch, it would still be installed. Now you can choose whether to install these dependencies or block them from being installed. For more information, see Create a patch baseline.</td>
</tr>
<tr>
<td>Create dynamic automation workflows with conditional branching</td>
<td>The <code>aws:branch</code> Automation action enables you to create a dynamic Automation workflow that evaluates multiple choices in a single step and then jumps to a different step in the Automation document based on the results of that evaluation. For more information, see Creating dynamic Automation workflows with conditional branching.</td>
</tr>
<tr>
<td>Use the AWS CLI to update Session Manager preferences</td>
<td>Instructions for using the CLI to update Session Manager preferences, such as CloudWatch Logs and S3 bucket logging options, have been added to the AWS Systems Manager User Guide. For information, see Use the AWS CLI to update Session Manager preferences.</td>
</tr>
<tr>
<td>Set up patching options more easily with the new 'Configure patching' page</td>
<td>Patch Manager has been updated with a new system for setting up patching configurations. On the Configure patching page, you can specify multiple patching options in a single location, including associating a maintenance window with a patching configuration and changing the patch baseline associated with a patch group. For more information, see About patching configurations and Create a patching configuration.</td>
</tr>
<tr>
<td>Updated SSM Agent requirement for Session Manager</td>
<td>Session Manager now requires SSM Agent version 2.3.68.0 or later. For more information about Session Manager prerequisites, see Complete Session Manager prerequisites.</td>
</tr>
<tr>
<td>Topic</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Manage instances without opening inbound ports or maintaining bastion hosts using Session Manager (p. 1201)</td>
<td>Now available, Session Manager is a fully managed AWS Systems Manager capability that lets you manage your EC2 instances through an interactive one-click browser-based shell or through the AWS CLI. Session Manager provides secure and auditable instance management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your EC2 instances. For more information, see Learn more about Session Manager.</td>
</tr>
<tr>
<td>Invoking other AWS services from a Systems Manager Automation workflow (p. 1201)</td>
<td>You can invoke other AWS services and other Systems Manager capabilities in your Automation workflow by using three new Automation actions (or plugins) in your Automation documents. For more information, see Invoking other AWS services from a Systems Manager Automation workflow.</td>
</tr>
<tr>
<td>Use Systems Manager-specific condition keys in IAM policies (p. 1201)</td>
<td>The topic Specifying conditions in a policy has been updated to list the IAM condition keys for Systems Manager that you can incorporate in policies. You can use these keys to specify the conditions under which a policy should take effect. The topic also includes links to example policies and other related topics.</td>
</tr>
<tr>
<td>Topic</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Aggregate inventory data with groups to see which instances are and aren’t configured to collect an inventory type (p. 1201)</td>
<td>Groups enable you to quickly see a count of which managed instances are and aren’t configured to collect one or more Inventory types. With groups, you specify one or more Inventory types and a filter that uses the <code>exists</code> operator. For more information, see Aggregating Inventory data.</td>
</tr>
<tr>
<td>View history and change tracking for Inventory and Configuration Compliance (p. 1201)</td>
<td>You can now view history and change tracking for Inventory collected from your managed instances. You can also viewing history and changing tracking for Patch Manager patching and State Manager associations reported by Configuration Compliance. For more information, see Viewing Inventory history and change tracking.</td>
</tr>
<tr>
<td>Systems Manager service-linked role extends support for maintenance window tasks (p. 1201)</td>
<td>The Maintenance Windows service requires a set of IAM permissions in order to run maintenance window tasks on your instances. Previously, the only option was to create a custom IAM role to supply these permissions. The service-linked role for Systems Manager has now been enhanced to provide these permissions, giving you two IAM role options. For more information, see Should I use a service-linked role or a custom service role to run maintenance window tasks?</td>
</tr>
<tr>
<td>Parameter Store integrates with Secrets Manager (p. 1201)</td>
<td>Parameter Store is now integrated with AWS Secrets Manager so that you can retrieve Secrets Manager secrets when using other AWS services that already support references to Parameter Store parameters. These services include Amazon EC2, Amazon Elastic Container Service, AWS Lambda, AWS CloudFormation, AWS CodeBuild, AWS CodeDeploy, and other Systems Manager capabilities. By using Parameter Store to reference Secrets Manager secrets, you create a consistent and secure process for calling and using secrets and reference data in your code and configuration scripts. For information, see Referencing AWS Secrets Manager secrets from Parameter Store parameters.</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Attach labels to Parameter Store parameters (p. 1201)</td>
<td>A parameter label is a user-defined alias to help you manage different versions of a parameter. When you modify a parameter, Systems Manager automatically saves a new version and increments the version number by one. A label can help you remember the purpose of a parameter version when there are multiple versions. For information, see Labeling parameters.</td>
</tr>
<tr>
<td>Create dynamic Automation workflows (p. 1201)</td>
<td>By default, the steps (or actions) that you define in the mainSteps section of an Automation document run in sequential order. After one action completes, the next action specified in the mainSteps section begins. With this release, you can now create Automation workflows that perform conditional branching. This means that you can create Automation workflows that dynamically respond to condition changes and jump to a specified step. For information, see Creating dynamic Automation workflows.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SSM Agent now pre-installed on Ubuntu Server 16.04 AMIs using Snap</td>
<td>Beginning with instances created from Ubuntu Server 16.04 AMIs identified with 20180627, the SSM Agent is pre-installed using Snap packages. On instances created from earlier AMIs, you should continue using deb installer packages. For information, see About SSM Agent installations on 64-bit Ubuntu Server 16.04 instances.</td>
</tr>
<tr>
<td>Review minimum S3 permissions required by SSM Agent</td>
<td>The new topic Minimum S3 bucket permissions for SSM Agent provides information about the Amazon Simple Storage Service (Amazon S3) buckets that resources might need to access to perform Systems Manager operations. You can specify these buckets in a custom policy if you want to limit S3 bucket access for an instance profile or VPC endpoint to the minimum required to use Systems Manager.</td>
</tr>
<tr>
<td>View complete execution history for a specific State Manager association ID</td>
<td>The new topic Viewing association histories describes how to view all executions for a specific association ID and then view execution details for one or more resources.</td>
</tr>
<tr>
<td>Patch Manager introduces support for Amazon Linux 2</td>
<td>You can now use Patch Manager to apply patches to Amazon Linux 2 instances. For general information about Patch Manager operating system support, see Patch Manager prerequisites. For information about the supported key-value pairs for Amazon Linux 2 when defining a patch filter, see PatchFilter in the AWS Systems Manager API Reference.</td>
</tr>
<tr>
<td>Send command output to Amazon CloudWatch Logs</td>
<td>The new topic Configuring Amazon CloudWatch Logs for Run Command describes how to send Run Command output to CloudWatch Logs.</td>
</tr>
</tbody>
</table>
### Earlier updates

The following table describes important changes in each release of the *AWS Systems Manager User Guide* before June 2018.

<table>
<thead>
<tr>
<th>Change Description</th>
<th>Details</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quickly create or delete Resource Data Sync for Inventory by using AWS CloudFormation (p. 1201)</td>
<td>You can use AWS CloudFormation to create or delete a Resource Data Sync for Systems Manager Inventory. To use AWS CloudFormation, add the <code>AWS::SSM::ResourceDataSync</code> resource to your AWS CloudFormation template. For more information, see Working with AWS CloudFormation Templates in the <em>AWS CloudFormation User Guide</em>. You can also manually create a Resource Data Sync for Inventory as described in Configuring Resource Data Sync for Inventory.</td>
<td>June 11, 2018</td>
</tr>
<tr>
<td>AWS Systems Manager User Guide update notifications now available through RSS (p. 1201)</td>
<td>The HTML version of the Systems Manager User Guide now supports an RSS feed of updates that are documented in the Systems Manager Documentation update history page. The RSS feed includes updates made in June, 2018, and later. Previously announced updates are still available in the Systems Manager documentation update history page. Use the RSS button in the top menu panel to subscribe to the feed.</td>
<td>June 6, 2018</td>
</tr>
<tr>
<td>Specify an exit code in scripts to reboot managed instances (p. 1201)</td>
<td>The new topic Rebooting managed instances from scripts describes how to instruct Systems Manager to reboot managed instances by specifying an exit code in scripts that you run with Run Command.</td>
<td>June 3, 2018</td>
</tr>
<tr>
<td>Create an event in Amazon CloudWatch Events whenever custom inventory is deleted (p. 1201)</td>
<td>The new topic Viewing inventory delete actions in CloudWatch Events describes how to configure Amazon CloudWatch Events to create an event anytime a user deletes custom Inventory.</td>
<td>June 1, 2018</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release date</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| Inventory all managed instances in your AWS account | You can easily inventory all managed instances in your AWS account by creating a global inventory association. For more information, see Inventory all managed instances in your AWS account (p. 719).  
  
  **Note**  
  Global inventory associations are available in SSM Agent version 2.0.790.0 or later. For information about how to update SSM Agent on your instances, see Update SSM Agent by using Run Command (p. 835). | May 3, 2018       |
<p>| SSM Agent installed by default on Ubuntu Server 18 | SSM Agent is installed, by default, on Ubuntu Server 18.04 LTS 64-bit and 32-bit AMIs.                                                                                                                       | May 2, 2018       |
| New topic                                  | The new topic Running commands using the document version parameter (p. 838) describes how to use the document-version parameter to specify which version of an SSM document to use when the command runs.                      | May 1, 2018       |
| New topic                                  | The new topic Deleting custom inventory (p. 738) describes how to delete custom inventory data from Amazon S3 by using the AWS CLI. The topic also describes how to use the SchemaDeleteOption to manage custom inventory by disabling or deleting a custom inventory type. This new feature uses the DeleteInventory API action. | April 19, 2018    |
| Amazon SNS notifications for SSM Agent      | You can subscribe to an Amazon SNS topic to receive notifications when a new version of SSM Agent is available. For more information, see Subscribe to SSM Agent notifications (p. 94).                                 | April 9, 2018     |
| CentOS patching support                     | Systems Manager now supports patching CentOS instances. For information about supported CentOS versions, see Patch Manager prerequisites (p. 912). For more information about how patching works, see How Patch Manager operations work (p. 913). | March 29, 2018    |
| New section                                | To provide a single source for reference information in the AWS Systems Manager User Guide, a new section has been introduced, AWS Systems Manager reference (p. 1189). Additional content will be added to this section as it becomes available. | March 15, 2018    |
| New topic                                  | The new topic About package name formats for approved and rejected patch lists (p. 952) details the package name formats you can enter in the lists of approved patches and rejected patches for a custom patch baseline. Sample formats are provided for each operating system type supported by Patch Manager. | March 9, 2018     |
| New topic                                  | Systems Manager now integrates with Chef InSpec. InSpec is an open-source, runtime framework that enables you to create human-readable profiles on GitHub or Amazon S3. Then you can use Systems Manager to run compliance scans and view compliant and noncompliant instances. For more | March 7, 2018     |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New topic</td>
<td>The new topic Using service-linked roles for Systems Manager (p. 1122) describes how to use an AWS Identity and Access Management (IAM) service-linked role with Systems Manager. Currently, service-linked roles are only required when using Systems Manager Inventory to collect metadata about tags and Resource Groups.</td>
<td>February 27, 2018</td>
</tr>
</tbody>
</table>
| New and updated topics      | You can now use Patch Manager to install patches that are in a different source repository than the default one configured on the instance. This is useful for patching instances with updates not related to security; with the content of Personal Package Archives (PPA) for Ubuntu Server; with updates for internal corporate applications; and so on. You specify alternative patch source repositories when you create a custom patch baseline. For more information, see the following topics:  
  • How to specify an alternative patch source repository (Linux) (p. 917)  
  • Create a custom patch baseline (p. 959)  
  • Create a patch baseline with custom repositories for different OS versions (p. 975)  

In addition, you can now use Patch Manager to patch SUSE Linux Enterprise Server instances. Patch Manager supports patching SLES 12.* versions (64-bit only). For more information, see the SLES-specific information in the following topics:  
  • How security patches are selected (p. 913)  
  • How patches are installed (p. 919)  
  • How patch baseline rules work on SUSE Linux Enterprise Server (p. 932) | February 6, 2018 |
<p>| New topic                   | The new topic Upgrade the Python requests module on Amazon Linux instances that use a proxy server (p. 88) provides instructions for ensuring that instances created using an Amazon Linux AMI have been updated with a current version of the Python requests module. This requirement is to ensure compatibility with Patch Manager. | January 12, 2018 |
| New topic                   | The new topic About SSM documents for patching instances (p. 935) describes the seven SSM documents currently available to help you keep your managed instances patched with the latest security-related updates. | January 10, 2018 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Important updates regarding Linux support</td>
<td>Updated various topics with the following information:</td>
<td>January 9, 2018</td>
</tr>
<tr>
<td></td>
<td>• SSM Agent is installed, by default, on Amazon Linux base AMIs dated 2017.09 and later.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• You must manually install SSM Agent on other versions of Linux, including non-base images like Amazon ECS-Optimized AMIs.</td>
<td></td>
</tr>
<tr>
<td>New topic</td>
<td>A new topic, <em>About the SSM document AWS-RunPatchBaseline (p. 938)</em>, provides details of how this SSM document operates on both Windows and Linux systems. It also provides information about the two available parameters in the <em>AWS-RunPatchBaseline</em> document, Operation and Snapshot ID.</td>
<td>January 5, 2018</td>
</tr>
<tr>
<td>New topics</td>
<td>A new section, <em>How Patch Manager operations work (p. 913)</em>, provides technical details that explain how Patch Manager determines which security patches to install and how it installs them on each supported operating system. It also provides information about how patch baseline rules work on different distributions of the Linux operating system</td>
<td>January 2, 2018</td>
</tr>
<tr>
<td>Retitled and moved the Systems Manager Automation Actions Reference</td>
<td>Based on customer feedback, the Automation Actions Reference is now called the Systems Manager Automation Document Reference. Furthermore, we moved the reference into the Shared Resources &gt; Documents node so it is closer to the <em>SSM document plugin reference (p. 1052)</em>. For more information, see <em>Systems Manager Automation actions reference (p. 386)</em>.</td>
<td>December 20, 2017</td>
</tr>
<tr>
<td>New Monitoring chapter and content</td>
<td>A new chapter, <em>Monitoring AWS Systems Manager (p. 1134)</em>, provides instructions for sending metrics and log data to Amazon CloudWatch Logs. A new topic, <em>Sending instance logs to CloudWatch Logs (CloudWatch agent) (p. 1135)</em>, provides instructions for migrating on-instance monitoring tasks, on 64-bit Windows Server instances only, from SSM Agent to the CloudWatch agent.</td>
<td>December 14, 2017</td>
</tr>
<tr>
<td>New chapter</td>
<td>A new chapter, <em>Identity and access management for AWS Systems Manager (p. 1105)</em>, provides comprehensive information about using AWS Identity and Access Management (IAM) and AWS Systems Manager to help secure access to your resources through the use of credentials. These credentials provide the permissions required to access AWS resources, such as accessing data stored in S3 buckets and sending commands to and reading the tags on EC2 instances.</td>
<td>December 11, 2017</td>
</tr>
<tr>
<td>Changes to the left navigation</td>
<td>We changed the headings in the left navigation of this user guide to match the headings in the new <em>AWS Systems Manager console</em>.</td>
<td>December 8, 2017</td>
</tr>
</tbody>
</table>
## Earlier updates

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiple changes for re:Invent 2017</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| • **Official launch of AWS Systems Manager:** AWS Systems Manager (formerly Amazon EC2 Systems Manager) is a unified interface that allows you to easily centralize operational data and automate tasks across your AWS resources. You can access the new AWS Systems Manager console [here](#). For more information, see [What is AWS Systems Manager?](#) (p. 1)  
• **YAML Support:** You can create SSM documents in YAML. For more information, see [AWS Systems Manager documents](#) (p. 1031). | November 29, 2017 |
| **Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes** | Using Run Command, you can take application-consistent snapshots of all [Amazon Elastic Block Store (Amazon EBS)](#) volumes attached to your Amazon EC2 Windows instances. The snapshot process uses the Windows [Volume Shadow Copy Service (VSS)](#) to take image-level backups of VSS-aware applications, including data from pending transactions between these applications and the disk. Furthermore, you don't need to shut down your instances or disconnect them when you need to back up all attached volumes. For more information, see [Using Run Command to Take VSS-Enabled Snapshots of EBS Volumes](#) in the [Amazon EC2 User Guide for Windows Instances](#). | November 20, 2017 |
| **Enhanced Systems Manager Security Available By Using VPC Endpoints** | You can improve the security posture of your managed instances (including managed instances in your hybrid environment) by configuring Systems Manager to use an interface VPC endpoint. Interface endpoints are powered by [PrivateLink](#), a technology that enables you to privately access Amazon EC2 and Systems Manager APIs by using private IP addresses. PrivateLink restricts all network traffic between your managed instances, Systems Manager, and EC2 to the Amazon network (managed instances don't have access to the Internet). Also, you don't need an Internet gateway, a NAT device, or a virtual private gateway. For more information, see [Optional] Create a Virtual Private Cloud endpoint](#) (p. 37). | November 7, 2017 |
### Inventory Support for Files, Services, Windows Roles, and the Windows Registry

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
</table>
| Inventory Support for Files, Services, Windows Roles, and the Windows Registry | SSM Inventory now supports gathering the following information from your managed instances.  
- **Files**: Name, size, version, installed date, modification and last accessed times, etc.  
- **Services**: Name, display name, status, dependent services, service type, start type, etc.  
- **Windows Registry**: Registry key path, value name, value type, and value.  
- **Windows roles**: Name, display name, path, feature type, installed state, etc. | November 6, 2017 |

Before you attempt to collect information for these inventory types, update SSM Agent on the instances you want to inventory. By running the latest version of SSM Agent, you ensure that you can collect metadata for all supported inventory types. For information about how to update SSM Agent by using State Manager, see [Automatically update SSM Agent (CLI)](p. 908).

For more information about Inventory, see [Learn more about Systems Manager Inventory](p. 707).

| Updates to Automation documentation | Fixed several issues in the information about setting up and configuring access for Systems Manager Automation. For more information, see [Getting started with Automation](p. 281). | October 31, 2017 |
### GitHub and Amazon S3 Integration

**Run remote scripts:** Systems Manager now supports downloading and running scripts from a private or public GitHub repository, and from Amazon S3. Using either the `AWS-RunRemoteScript` pre-defined SSM document or the `aws:downloadContent` plugin in a custom SSM document, you can run Ansible Playbooks and scripts in Python, Ruby, or PowerShell, to name a few. These changes further enhance *infrastructure as code* when you use Systems Manager to automate configuration and deployment of EC2 instances and on-premises managed instances in your hybrid environment. For more information, see Partner and product integration (p. 98).

**Create composite SSM documents:** Systems Manager now supports running one or more secondary SSM documents from a primary SSM document. These primary documents that run other documents are called *composite* documents. Composite documents enable you to create and share a standard set of secondary SSM documents across AWS accounts for common tasks such as boot-strapping anti-virus software or domain-joining instances. You can run composite and secondary documents stored in Systems Manager, GitHub, or Amazon S3. After you create a composite document, you can run it by using the `AWS-RunDocument` pre-defined SSM document. For more information, see Creating composite documents (p. 1089) and Running SSM documents from remote locations (p. 1099).

**SSM document plugin reference:** For easier access, we moved the SSM Plugin Reference for SSM documents out of the Systems Manager API Reference and into the User Guide. For more information, see SSM document plugin reference (p. 1052).

### Support for Parameter Versions in Parameter Store

When you edit a parameter, Parameter Store now automatically iterates the version number by 1. You can specify a parameter name and a specific version number in API calls and SSM documents. If you don't specify a version number, the system automatically uses the latest version.

Parameter versions provide a layer of protection in the event that a parameter is accidentally changed. You can view the values of all versions, and reference older versions if necessary. You can also use parameter versions to see how many times a parameter changed over a period of time. For more information, see Working with parameter versions (p. 259).

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub and Amazon S3 Integration</td>
<td><strong>Run remote scripts:</strong> Systems Manager now supports downloading and running scripts from a private or public GitHub repository, and from Amazon S3. Using either the <code>AWS-RunRemoteScript</code> pre-defined SSM document or the <code>aws:downloadContent</code> plugin in a custom SSM document, you can run Ansible Playbooks and scripts in Python, Ruby, or PowerShell, to name a few. These changes further enhance <em>infrastructure as code</em> when you use Systems Manager to automate configuration and deployment of EC2 instances and on-premises managed instances in your hybrid environment. For more information, see Partner and product integration (p. 98). <strong>Create composite SSM documents:</strong> Systems Manager now supports running one or more secondary SSM documents from a primary SSM document. These primary documents that run other documents are called <em>composite</em> documents. Composite documents enable you to create and share a standard set of secondary SSM documents across AWS accounts for common tasks such as boot-strapping anti-virus software or domain-joining instances. You can run composite and secondary documents stored in Systems Manager, GitHub, or Amazon S3. After you create a composite document, you can run it by using the <code>AWS-RunDocument</code> pre-defined SSM document. For more information, see Creating composite documents (p. 1089) and Running SSM documents from remote locations (p. 1099).</td>
<td>October 26, 2017</td>
</tr>
<tr>
<td>Support for Parameter Versions in Parameter Store</td>
<td>When you edit a parameter, Parameter Store now automatically iterates the version number by 1. You can specify a parameter name and a specific version number in API calls and SSM documents. If you don't specify a version number, the system automatically uses the latest version. Parameter versions provide a layer of protection in the event that a parameter is accidentally changed. You can view the values of all versions, and reference older versions if necessary. You can also use parameter versions to see how many times a parameter changed over a period of time. For more information, see Working with parameter versions (p. 259).</td>
<td>October 24, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release date</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td><strong>Support for Tagging Systems Manager Documents</strong> You can now use the AddTagsToResource API, the AWS CLI, or the AWS Tools for PowerShell to tag Systems Manager documents with key-value pairs. Tagging helps you quickly identify specific resources based on the tags you’ve assigned to them. This is in addition to existing tagging support for managed instances, maintenance windows, Parameter Store parameters, and patch baselines. For information, see Tagging Systems Manager documents (p. 1164).</td>
<td>October 3, 2017</td>
</tr>
</tbody>
</table>
|        | **Various Documentation Updates to Fix Errors or Update Content Based on Feedback**  
|        | • Updated Setting up AWS Systems Manager for hybrid environments (p. 43) with information for Raspbian Linux.  
|        | • Updated Setting up AWS Systems Manager (p. 25) with new requirement for Windows Server instances. SSM Agent requires Windows PowerShell 3.0 or later to run certain SSM Documents on Windows Server instances (for example, the legacy AWS-ApplyPatchBaseline SSM document). Verify that your Windows Server instances are running Windows Management Framework 3.0 or later. The framework includes PowerShell. For more information, see Windows Management Framework 3.0. | October 2, 2017 |
|        | **Troubleshoot Unreachable Windows Instances by Using the EC2Rescue Automation Workflow** EC2Rescue can help you diagnose and troubleshoot problems on Amazon EC2 Windows Server instances. You can run the tool as a Systems Manager Automation workflow by using the AWSSupport-ExecuteEC2Rescue document. The AWSSupport-ExecuteEC2Rescue document is designed to perform a combination of Systems Manager actions, AWS CloudFormation actions, and Lambda functions that automate the steps normally required to use EC2Rescue. For more information, see Walkthrough: Run the EC2Rescue tool on unreachable instances (p. 591). | September 29, 2017 |
|        | **SSM Agent Installed By Default on Amazon Linux** SSM Agent is installed, by default, on Amazon Linux AMIs dated 2017.09 and later. You must manually install SSM Agent on other versions of Linux, as described in Installing and configuring SSM Agent on EC2 instances for Linux (p. 69). | September 27, 2017 |
|        | **Run Command Enhancements** Run Command includes the following enhancements.  
|        | • You can restrict command execution to specific instances by creating an IAM user policy that includes a condition that the user can only run commands on instances that are tagged with specific Amazon EC2 tags. For more information, see Restricting Run Command access based on instance tags (p. 831).  
<p>|        | • You have more options for targeting instances by using Amazon EC2 tags. You can now specify multiple tag keys and multiple tag values when sending commands. For more information, see Using targets and rate controls to send commands to a fleet (p. 840). | September 12, 2017 |
|        | <strong>Systems Manager Supported on Raspbian</strong> Systems Manager can now run on Raspbian Jessie and Raspbian Stretch devices, including Raspberry Pi (32-Bit). For more information, see Manually install SSM Agent on Raspbian instances (p. 77). | September 7, 2017 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Automatically Send SSM Agent Logs to Amazon CloudWatch Logs</strong></td>
<td>You can now make a simple configuration change on your instances to have SSM Agent send log files to CloudWatch. For more information, see [Sending instance logs to CloudWatch Logs (SSM Agent)](p. 1141).</td>
<td>September 7, 2017</td>
</tr>
<tr>
<td><strong>Encrypt Resource Data Sync</strong></td>
<td>Systems Manager Resource Data Sync lets you aggregate inventory data collected on dozens or hundreds of managed instance in a central S3 bucket. You can now encrypt Resource Data Sync by using an AWS Key Management Service key. For more information, see [Walkthrough: Use Resource Data Sync to aggregate inventory data](p. 751).</td>
<td>September 1, 2017</td>
</tr>
</tbody>
</table>
| **New State Manager Walkthroughs** | Added two new walkthroughs to the State Manager documentation:  
*Automatically update SSM Agent (CLI)* (p. 908)  
*Walkthrough: Automatically update PV drivers on EC2 instances for Windows Server (console)* (p. 910) | August 31, 2017 |
<p>| <strong>Systems Manager Configuration Compliance</strong> | Use Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant. By default, Configuration Compliance displays compliance data about Patch Manager patching and State Manager associations. You can also customize the service and create your own compliance types based on your IT or business requirements. For more information, see [AWS Systems Manager Configuration Compliance](p. 696). | August 28, 2017 |
| <strong>New Automation Action:</strong> aws:executeAutomation | Runs a secondary Automation workflow by calling a secondary Automation document. With this action, you can create Automation documents for your most common workflows, and reference those documents during an Automation execution. This action can simplify your Automation documents by removing the need to duplicate steps across similar documents. For more information, see [aws:executeAutomation – Run another automation execution](p. 413). | August 22, 2017 |
| <strong>Automation as the Target of a CloudWatch Event</strong> | You can start an Automation workflow by specifying an Automation document as the target of an Amazon CloudWatch event. You can start workflows according to a schedule, or when a specific AWS system event occurs. For more information, see [Running Automation workflows with triggers using CloudWatch Events](p. 314). | August 21, 2017 |
| <strong>State Manager Association Versioning and General Updates</strong> | You can now create different State Manager association versions. There is a limit of 1,000 versions for each association. You can also specify names for your associations. Also, the State Manager documentation has been updated to address outdated information and inconsistencies. For more information, see [AWS Systems Manager State Manager](p. 864). | August 21, 2017 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
</table>
| Changes to Maintenance Windows | Maintenance Windows include the following changes or enhancements:  
- Previously, Maintenance Windows could only perform tasks by using Run Command. You can now perform tasks by using Systems Manager Automation, AWS Lambda, and AWS Step Functions.  
- You can edit the targets of a maintenance window, specify a target name, description, and owner.  
- You can edit tasks in a maintenance window, including specifying a new SSM document for Run Command and Automation tasks.  
- All Run Command parameters are now supported, including DocumentHash, DocumentHashType, TimeoutSeconds, Comment, and NotificationConfig.  
- You can now use a `safe` flag when you attempt to deregister a target. If enabled, the system returns an error if the target is referenced by any task. | August 16, 2017 |
| New Automation Action: aws:approve | This new action for Automation documents temporarily pauses an Automation execution until designated principals either approve or reject the action. After the required number of approvals is reached, the Automation execution resumes. | August 10, 2017 |
| Automation Assume Role No Longer Required | Automation previously required that you specify a service role (or assume role) so that the service had permission to perform actions on your behalf. Automation no longer requires this role because the service now operates by using the context of the user who invoked the execution.  
However, the following situations still require that you specify a service role for Automation:  
- When you want to restrict a user’s privileges on a resource, but you want the user to run an Automation workflow that requires elevated privileges. In this scenario, you can create a service role with elevated privileges and allow the user to run the workflow.  
- Operations that you expect to run longer than 12 hours require a service role. | August 3, 2017 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Compliance</td>
<td>Use Amazon EC2 Systems Manager Configuration Compliance to scan your fleet of managed instances for patch compliance and configuration inconsistencies. You can collect and aggregate data from multiple AWS accounts and Regions, and then drill down into specific resources that aren’t compliant. For more information, see AWS Systems Manager Configuration Compliance (p. 696).</td>
<td>August 8, 2017</td>
</tr>
</tbody>
</table>
| SSM Document Enhancements | SSM Command and Policy documents now offer cross-platform support. This means that a single SSM document can process plugins for Windows and Linux operating systems. Cross-platform support enables you to consolidate the number of documents you manage. Cross-platform support is offered in SSM documents that use schema version 2.2 or later.  

SSM Command documents that use schema version 2.0 or later can now include multiple plugins of the same type. For example, you can create a Command document that calls the aws:runRunShellScript plugin multiple times.  

For more information about schema version 2.2 changes, see AWS Systems Manager documents (p. 1031). For more information about SSM plugins, see Systems Manager Plugins.                                                                                   | July 12, 2017    |
| Linux Patching            | Patch Manager can now patch the following Linux distributions:  

**64-bit and 32-bit systems**  
- Amazon Linux 2014.03, 2014.09, or later  
- Ubuntu Server 16.04 LTS, 14.04 LTS, or 12.04 LTS  
- Red Hat Enterprise Linux (RHEL) 6.5 or later  

**64-bit systems only**  
- Amazon Linux 2015.03, 2015.09, or later  
- Red Hat Enterprise Linux (RHEL) 7.x or later  

For more information, see AWS Systems Manager Patch Manager (p. 911).  

**Note**  
- To patch Linux instances, your instances must be running SSM Agent version 2.0.834.0 or later. For information about updating the agent, see the section titled Example: Update SSM Agent in Running commands from the console (p. 834).  
- The AWS-ApplyPatchBaseline SSM document is being replaced by the AWS-RunPatchBaseline document.                                                                                                                   | July 6, 2017     |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Data Sync</td>
<td>You can use Systems Manager Resource Data Sync to send Inventory data collected from all of your managed instances to a single Amazon S3 bucket. Resource Data Sync then automatically updates the centralized data when new Inventory data is collected. With all Inventory data stored in a target S3 bucket, you can use services like Amazon Athena and Amazon QuickSight to query and analyze the aggregated data. For more information, see Configuring Resource Data Sync for Inventory (p. 712). For an example of how to work with Resource Data Sync, see Walkthrough: Use Resource Data Sync to aggregate inventory data (p. 751).</td>
<td>June 29, 2017</td>
</tr>
</tbody>
</table>
| Systems Manager Parameter Hierarchies | Managing dozens or hundreds of Systems Manager parameters as a flat list is time-consuming and prone to errors. You can use parameter hierarchies to help you organize and manage Systems Manager parameters. A hierarchy is a parameter name that includes a path that you define by using forward slashes. Here is an example that uses three hierarchy levels in the name to identify the following:  

/Environment/Type of computer/Application/Data  

/Dev/DBServer/MySQL/db-string13  

For more information, see Organizing parameters into hierarchies (p. 240). For an example of how to work with parameter hierarchies, see Walkthrough: Manage parameters using hierarchies (AWS CLI) (p. 274). | June 22, 2017 |
| SSM Agent Support for SUSE Linux Enterprise Server | You can install SSM Agent on 64-bit SUSE Linux Enterprise Server (SLES). For more information, see Installing and configuring SSM Agent on EC2 instances for Linux (p. 69). | June 14, 2017 |
AWS glossary

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.